TAB 1

US006192508B1

(12) **United States Patent**
Malik et al.

(10) **Patent No.: US 6,192,508 B1**
(45) **Date of Patent: Feb. 20, 2001**

(54) **METHOD FOR LOGIC OPTIMIZATION FOR IMPROVING TIMING AND CONGESTION DURING PLACEMENT IN INTEGRATED CIRCUIT DESIGN**

(75) Inventors: **Sharad Malik**, Princeton, NJ (US); **Lawrence Pileggi**, Pittsburgh, PA (US); **Abhijeet Chakraborty**, Sunnyvale, CA (US); **Gary K. Yeap**, San Jose, CA (US); **Douglas B. Boyle**, Palo Alto, CA (US)

(73) Assignee: **Monterey Design Systems**, Sunneyvale, CA (US)

(*) Notice: Under 35 U.S.C. 154(b), the term of this patent shall be extended for 0 days.

(21) Appl. No.: **09/097,076**

(22) Filed: **Jun. 12, 1998**

(51) **Int. Cl.**[7] **G06F 17/50**
(52) **U.S. Cl.** **716/9**; 716/10; 716/13
(58) **Field of Search** 395/500.1; 716/9, 716/10, 13

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,484,292 * 11/1984 Hong et al. 716/13
5,557,533 * 9/1996 Koford et al. 706/13
5,561,772 * 10/1996 Dorner et al. 710/101
5,572,482 * 11/1996 Hoshizaki et al. 365/233
5,847,965 * 12/1998 Cheng 395/500.09

* cited by examiner

Primary Examiner—Kevin J. Teska
Assistant Examiner—Hugh Jones
(74) *Attorney, Agent, or Firm*—Burns Doane Swecker & Mathis L.L.P

(57) **ABSTRACT**

This invention recognizes the ability of logic optimization to help placement relieve congestion. Different types of logic optimizations are used to help placement relieve congestion. In one type of optimization, the speed of parts of the circuit is improved by selecting faster cells. In another type of optimization, the topology of the circuit is changed such that placement can now move cells, which could not have been moved before, to reduce congestion and thus enable routing. A distinguishing feature of this methodology is that it not only uses the placement information for interconnection delay/area estimates during logic optimization, but also uses logic optimization to aid the physical placement steps by providing support to placement so that the congestion of the circuit is improved. The aim is to avoid getting into a situation where the placed circuit cannot be routed.

18 Claims, 4 Drawing Sheets

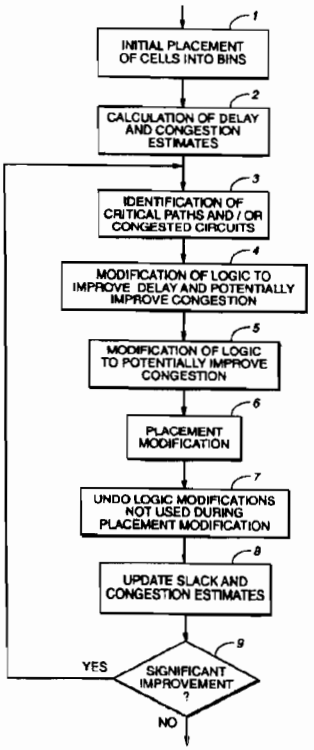


FIG._1

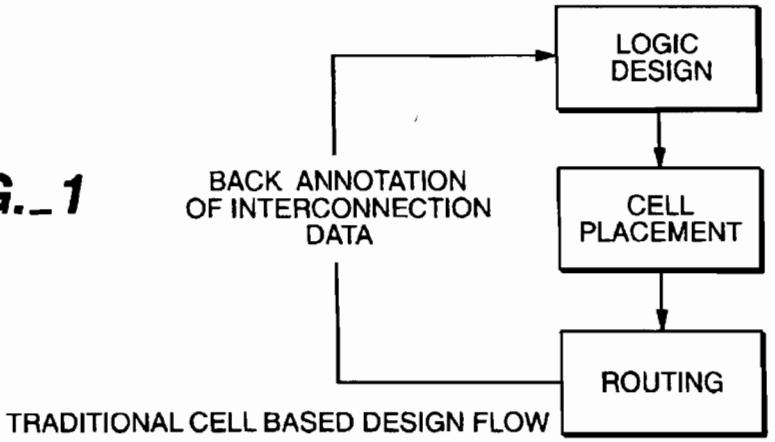


FIG._3A

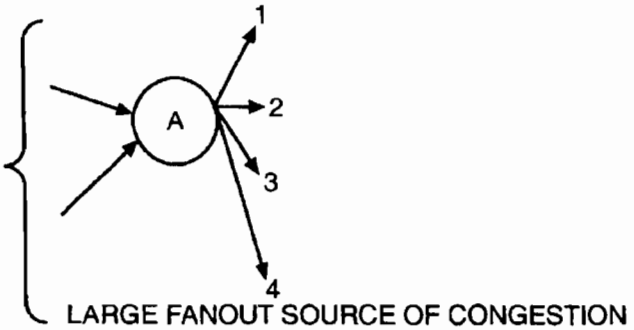


FIG._3B

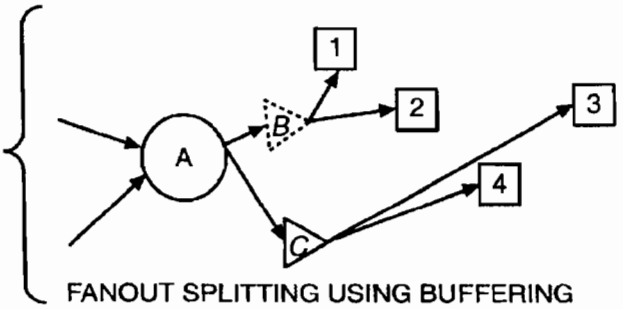
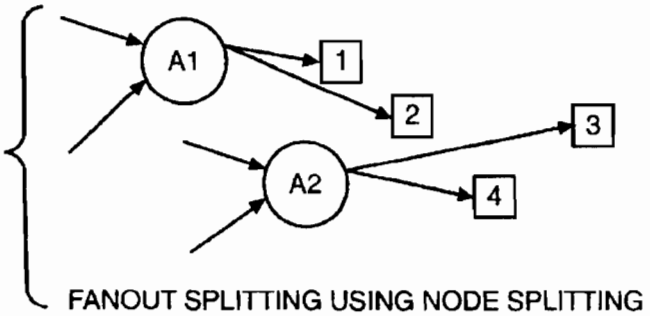


FIG._3C



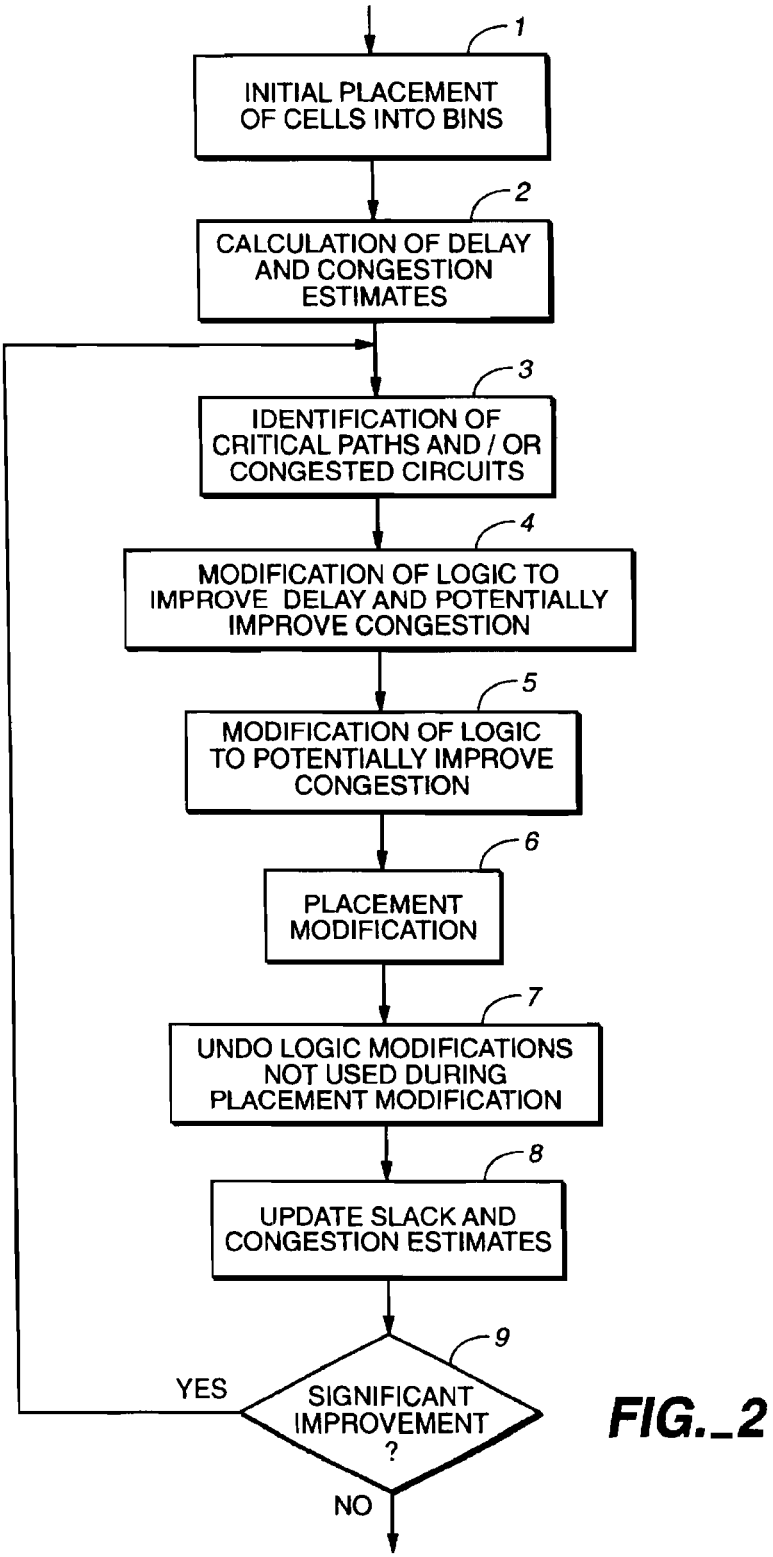
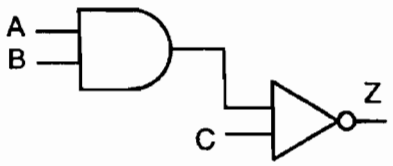

1
INITIAL PLACEMENT OF CELLS INTO BINS
2
CALCULATION OF DELAY AND CONGESTION ESTIMATES
3
IDENTIFICATION OF CRITICAL PATHS AND / OR CONGESTED CIRCUITS
4
MODIFICATION OF LOGIC TO IMPROVE DELAY AND POTENTIALLY IMPROVE CONGESTION
5
MODIFICATION OF LOGIC TO POTENTIALLY IMPROVE CONGESTION
6
PLACEMENT MODIFICATION
7
UNDO LOGIC MODIFICATIONS NOT USED DURING PLACEMENT MODIFICATION
8
UPDATE SLACK AND CONGESTION ESTIMATES
9
SIGNIFICANT IMPROVEMENT ?
YES
NO

FIG._2



BOTH GATES ARE IN THE SAME BIN

FIG._4A

FIG._4B

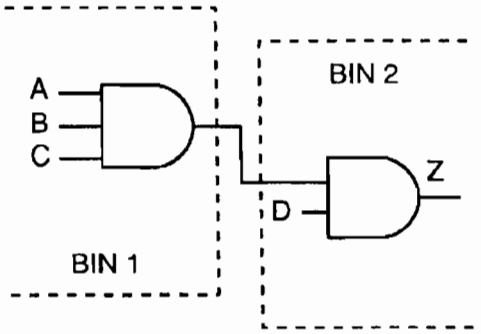


FIG._5A

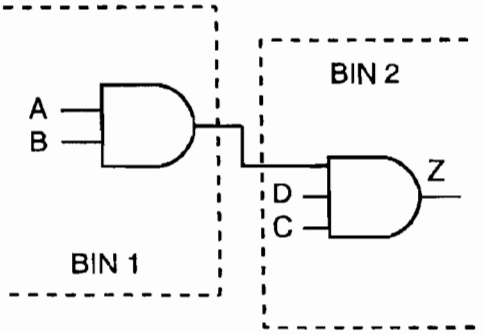


FIG._5B

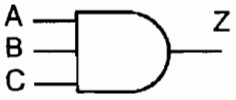


FIG._6A

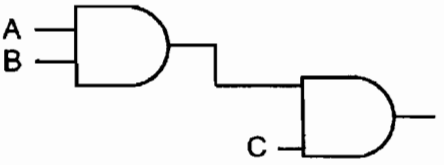


FIG._6B

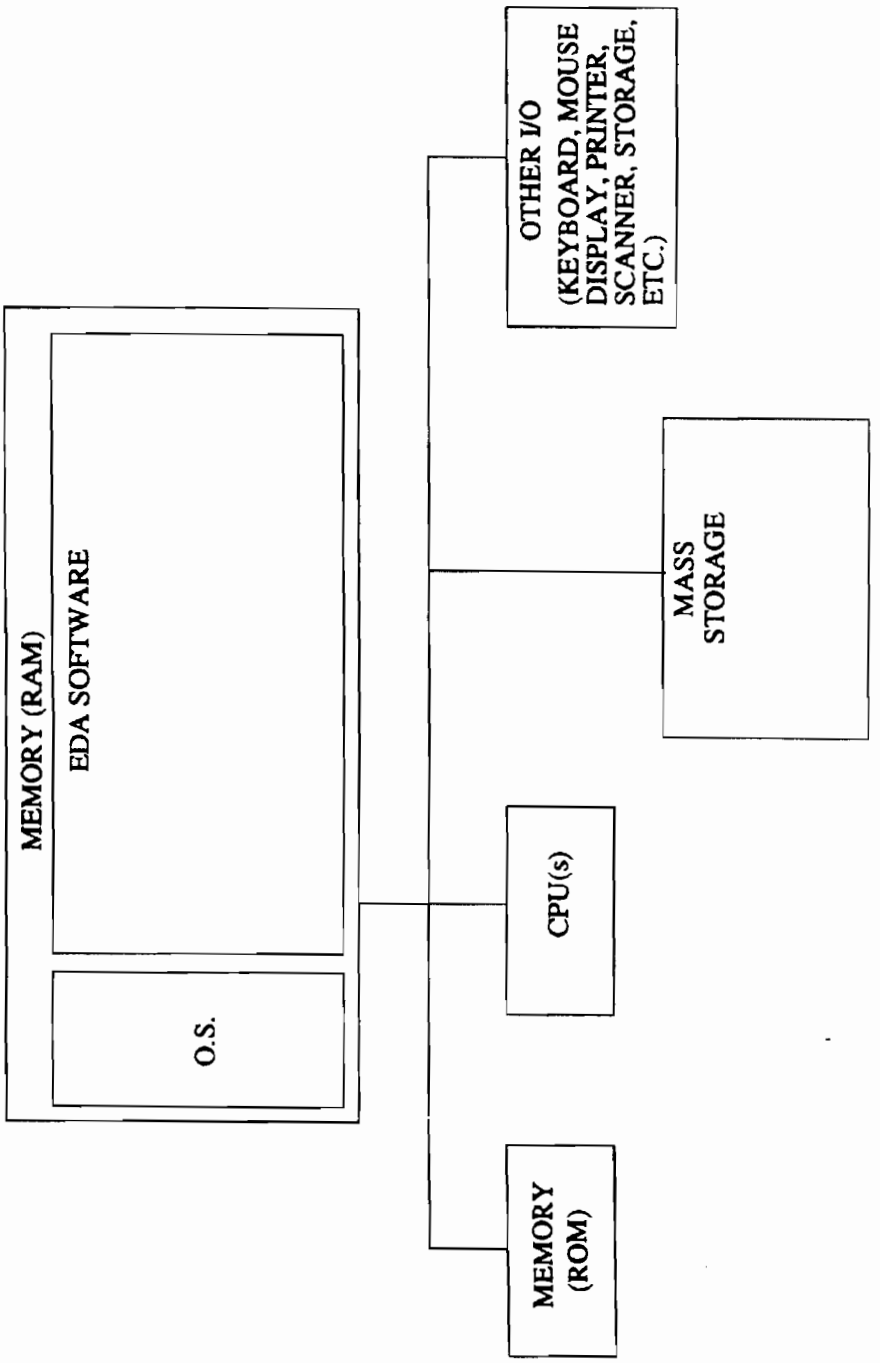

MEMORY (RAM)
EDA SOFTWARE
O.S.
OTHER I/O (KEYBOARD, MOUSE DISPLAY, PRINTER, SCANNER, STORAGE, ETC.)
MASS STORAGE
CPU(s)
MEMORY (ROM)

Fig. 7

METHOD FOR LOGIC OPTIMIZATION FOR IMPROVING TIMING AND CONGESTION DURING PLACEMENT IN INTEGRATED CIRCUIT DESIGN

This application is related by subject matter to U.S. Application Ser. No. 09/097,299 entitled METHOD FOR DESIGN OPTIMIZATION USING LOGICAL AND PHYSICAL INFORMATION, filed on even date herewith and incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to integrated circuit design and layout.

2. State of the Art

Traditional cell-based integrated circuit design follows several steps. The first step is designing the logical gate-level circuit that implements the function to be realized by the circuit (referred to as logic design or logic synthesis, of which logic optimizations are a key part). The next step is placing the gates (or cells) in a physical layout, and the final step is routing the interconnection between the cells. With increasing dominance of interconnection delays and area in circuits implemented in deep submicron technologies, this approach is proving to be no longer viable. The problem is that, during the logic optimization stage, the interconnection is not known yet, and thus the dominant part of the area and the delay cannot be considered.

Attempts to overcome this problem have considered alternating logic synthesis and placement and routing, with "back annotation" of the interconnect information to the subsequent logic synthesis steps. Referring to FIG. 1, showing traditional cell-based design flow, a logic design phase is followed by a cell placement phase and then a routing phase. Following the routing phase, interconnection data is back annotated. The logic design, cell placement and routing phases are then repeated. This cycle is continued until, during the routing phase, the design is successfully routed. The problem with this method is that the logic synthesis steps that consider the back annotation information cannot guarantee to fix problems that prevent routing without introducing additional problems due to the modifications made to the circuit gates and topology. There results a large number of iterations between logic synthesis and subsequent place and route, with the possibility of the process never converging.

An alternative approach is to consider placement information during logic optimization. In this methodology, sometimes termed "placement aware synthesis," placement information is made available in varying degrees during logic optimization, i.e. some placement is done as part of logic synthesis (sometimes referred to in the industry as just synthesis). Logic optimization uses this placement information to estimate the effect of the interconnects on the delay and the area of the circuit. Thus logic optimization attempts to accurately model the interconnect delay and area that might result during a placement step. However, it may result in a placed circuit that cannot be routed using the area resources provided by the placement step. The inability to route the resulting placed circuit results in modifications to the placement, consequently nullifying the interconnection information used during logic optimization.

A circuit that has been placed but cannot be routed subject to the available area constraints is not realizable. Additional routing resources must be created to enable the routing.

There results an increase in circuit area and possibly delay, since the wires may now need to go through longer paths.

Placement algorithms are limited in how they can place cells by the timing constraints placed on the design. The timing constraints may result in certain parts of the design being very congested in terms of the wiring (or interconnection) resources needed to connect the cells in those parts of the circuit. It would be possible to relieve the congestion if somehow the cells in the congested area were to be moved apart. However, moving the cells apart may result in an increase in the interconnection delays, which in turn may result in a violation of the timing constraints. Thus a situation results where it is possible to have acceptable timing slacks or acceptable congestion but not both.

The paper by Villarubia and Hojat (ICCD 97) proposes integrated logic optimization and placement. However, the proposed methodology alternates placement and logic optimization and does not consider the impact of the logic optimizations on subsequent placement steps.

SUMMARY OF THE INVENTION

This invention recognizes the ability of logic optimization to help placement relieve congestion. Different types of logic optimizations are used to help placement relieve congestion. In one type of optimization, the speed of parts of the circuit is improved by selecting faster cells. In another type of optimization, the topology of the circuit is changed such that placement can now move cells, which could not have been moved before, to reduce congestion and thus enable routing. A distinguishing feature of this methodology is that it not only uses the placement information for interconnection delay/area estimates during logic optimization, but also uses logic optimization to aid the physical placement steps by providing support to placement so that the congestion of the circuit is improved. The aim is to avoid getting into a situation where the placed circuit cannot be routed.

There are two specific ways in which logic optimization aids placement in relieving congestion. The first method involves determining parts of the circuit which are congested, and then speeding up the logic in these parts. This speedup provides timing slack for a subsequent placement step to move cells while ensuring that this move does not cause the modified interconnections to violate timing constraints. The second method involves modifying the topology of the circuit by adding gates while maintaining the functionality, such that the added gates can then be moved by the placement steps to relieve congestion.

An important aspect of the optimizations, specifically directed towards helping placement relieve congestion, is the ability to undo modifications if placement does not actually use the modifications. The undo capability ensures that no area/power resources are wasted for transformations that are not used as intended.

A critical problem in using logic optimization as part of placement is that logic optimization steps can and do increase the area of circuits. This increase in area can invalidate the results of any placement done thus far, and consequently result in the inability of the combination of these steps to converge. An important part of this invention is to actively bound the area increase of specific parts of the circuit which guarantees that the current placement results are still valid after the logic optimizations, consequently guaranteeing convergence of the integrated logic optimization and placement steps.

BRIEF DESCRIPTION OF THE DRAWING

The present invention may be further understood from the following description in conjunction with the appended drawing. In the drawing:

FIG. **1** is a flowchart of traditional cell-based design flow;

FIG. **2** is a flowchart of design flow in accordance with the present invention;

FIG. **3**(*a*) is a diagram of a gate having a large fanout;

FIG. **3**(*b*) is a diagram of the gate of FIG. **3**(*a*) following fanout splitting using buffering;

FIG. **3**(*c*) is a diagram of a circuit equivalent to the gate of FIG. **3**(*a*) following fanout splitting using node splitting;

FIG. **4**(*a*) is a diagram of a circuit to which intra-bin pin density logic optimization may be applied;

FIG. **4**(*b*) is a diagram of an equivalent circuit resulting from intra-bin pin density logic optimization applied to the circuit of FIG. **4**(*a*);

FIG. **5**(*a*) is a diagram of a circuit to which inter-bin pin density logic optimization may be applied;

FIG. **5**(*b*) is a diagram of an equivalent circuit resulting from inter-bin pin density logic optimization applied to the circuit of FIG. **5**(*a*);

FIG. **6**(*a*) is a diagram of a circuit to which input splitting logic optimization may be applied;

FIG. **6**(*b*) is a diagram of an equivalent circuit resulting from input splitting logic optimization applied to the circuit of FIG. **6**(*a*);

FIG. **7** is a block diagram of a computer system that may be used to practice the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention may be used in conjunction with an electronic design automation placement tool. In accordance with an exemplary embodiment of one such placement tool, at each stage in cell placement, the cells are partitioned into a number of bins. Interconnection models for interconnects between bins and within bins provide both delay estimates for each interconnect in the circuit, as well as congestion estimates for each bin in the circuit. The circuit has timing constraints imposed on it that it needs to satisfy. The delay estimates of the interconnection, combined with the delays of the cells and the timing constraints imposed on the design, are converted to timing slack information for each part of the circuit. A negative timing slack indicates that that part of the circuit is not meeting the timing constraints. A positive slack indicates that that part of the circuit is producing its result faster than is needed and can thus be slowed down without violating its timing constraints. More generally, "slack" is defined herein as a measure of the degree to which a timing requirement is met in an integrated circuit design.

The traditional role of logic synthesis has been to identify areas of the circuit which have negative timing slack and then modify the circuit so as to fix this problem. As described herein, logic synthesis is used to aid placement to achieve both acceptable delays and congestion, by making circuit modifications that increase the timing slack in the congested parts. Referring more particularly to FIG. **2**, the steps involved in this process are, in general, as follows:

Initial placement of cells into bins (Step 1).

Calculation of delay estimates, i.e., slack estimates, and congestion estimates (Step 2).

Identification of critical paths and/or congested circuits (Step 3). In the case of congested circuits, identification of cells to be modified for in order for placement moves to relieve congestion.

Modification of logic to improve delay (Step 4), e.g., speeding up part of the circuit to improve slack in that part of the circuit. Conventional logic optimization techniques such as remapping and buffering are used for this. The purpose of this step is twofold. Such timing improvement is desirable in and of itself Also, if positive slack is achieved for parts of the congested circuit, this positive slack provides room for a subsequent placement step to move the cells in this part further away to reduce congestion.

Modification of logic to potentially improve circuit congestion (Step 5). Techniques such as fanout splitting are used for this.

Placement modification to take advantage of the preceding modifications (Step 6).

Undo logic modifications not used in the preceding placement modifications (Step 7).

Update slack and congestion estimates (Step 8).

Repeat for so long as significant improvement is obtained (Step 9).

Note that in various embodiments of the invention, not all of the foregoing steps may be practiced and that the order of the steps practiced may vary from the order of steps as presented above.

Particular logic modifications used to relieve congestion will be described in greater detail. Placement algorithms are limited in how they can place cells by the topology of the circuit. If the output of cell A is connected to (also referred to as "fanning out to") four different terminals in different cells (indicated by the numbers **1**–**4**) in FIG. **3**(*a*), then the placement of A is strongly influenced by the placement of cells corresponding to these terminals. In addition, because the output of A needs to be routed to four different places, the output of A is likely to cause congestion in this part of the circuit. Modifying the circuit topology without changing the logic functionality can avoid the bunching of wires at the output of A. This general step is referred to as fanout splitting. There are two distinct ways in which fanout splitting is done.

The first method involves buffering and is illustrated in FIG. **3**(*b*). Here buffers B and C are added such that B is used to drive terminals **1** and **2** and C is used to drive **3** and **4**. The grouping of terminals and assignments to buffers is done using geometric proximity of the terminals. Once the fanouts have been distributed between the buffers, a subsequent placement step can now move the buffers closer to the terminal they are connected to, relieving congestion due to the large fanout at the output of A.

In FIG. **3**(*c*) an alternative technique is used. Two copies of node A are used, labeled **A1** and **A2**, with **A1** fanning out to 1 and 2, and **A2** fanning out to 3 and 4. This technique is referred to as node splitting. Once node splitting is done a subsequent placement step can move **A1** or **A2** closer to the terminals they are connected to, in order to relieve congestion.

To summarize, the steps involved in fanout splitting are:

Identification of congested bins. This is done using the congestion estimates for each bin.

Identification of large fanout cells resulting in congestion.

Modification of the circuit topology using fanout splitting by either buffering or node splitting.

Further examples of logic modifications that may be used to relieve congestion will now be described.

One measure of the congestion in a bin is given by pin density, calculated as the total number of pins in the bin divided by the total routable area in the bin. Here a pin refers to either an input or an output of a cell. It is desirable to get a lower congestion since that is likely to make routing easier. It is possible for logic optimizations to directly reduce this measure of congestion.

Intra-bin pin density logic optimization is done by replacing a set of gates in a bin with a different but logically equivalent set. Referring to FIG. 4(*a*), the AND gate followed by the NOR gate is logically equivalent to the AND-OR-INVERT gate shown in FIG. 4(*b*). In this case assume that the total routable area is the same before and after the logic change. However, the AND-OR-INVERT gate in FIG. 4(*b*) has fewer pins (4) compared to the AND and the NOR gates (3 each for a total of 6 pins) in FIG. 4(*a*). Intuitively, elimination of the extra net between the AND and the NOR gate in FIG. 4(*a*) will make the bin less congested.

Pin density can be reduced in a congested bin by possibly increasing it in a less congested bin. This technique is referred to as inter-bin logic optimization. FIG. 5(*a*) shows two AND gates in different bins. Assume that Bin 1 is over congested and Bin 2 is undercongested. By using the associative property of AND gates, a connection (C) can be moved from the AND gate in Bin 1 to that in Bin 2 as shown in FIG. 5(*b*). This reduces the pin density in Bin 1 (the number of pins is reduced from 4 to 3) and thus reduces congestion. Note that the pin density and thus the congestion in Bin 2 has increased in the process (the number of pins increases from 3 to 4), but that is acceptable since Bin 2 was undercongested.

Another logic optimization technique is input splitting. The motivation for this technique is similar to that for fanout splitting. A gate with a large number of input pins is replaced by a set of gates each one of which has a smaller number of input pins. While this may increase the pin density, it provides flexibility for a subsequent placement step to move some of these gates from an over congested bin to an undercongested bin in order to improve congestion.

FIG. 4(*b*) shows an AND-OR-INVERT gate with three inputs. Input splitting results in this gate being replaced by the an AND gate followed by a NOR gate as in FIG. 4(*a*). While this may result in increasing the pin density in the bin, it allows a subsequent placement step to move either of the two gates into a different undercongested bin.

FIG. 6(*a*) shows a three input AND gate. Input splitting results in this being replaced by two, two input AND gates as shown in FIG. 6(*b*). A subsequent placement step may now move either of these gates to a different undercongested bin.

For many of the congestion relieving logic synthesis methods proposed as part of placement, there are two important issues that this invention addresses. In most cases, logic synthesis cannot itself improve congestion, but rather only provide opportunities for placement to improve congestion, it is important to track which of these opportunities are actually used. Any unused opportunities may result in wasted resources, since the logic optimization step used to create them typically uses additional area and power (for faster cells) or additional gates. The use of the logic optimizations during placement is therefore actively tracked. Any unused optimizations are undone to ensure that there are no wasted resources.

It is important that the area used by the logic optimizations be monitored. Because the current placement (at the time of the logic optimizations) is based on a certain area of all the bins, if this information changes, then the placement may no longer be appropriate. The change may result in placement being done again at that step, and possibly the process never converging. Monitoring of the area used in order to preserve the feasibility of the placement is done by placing an upper bound on the area of each bin. The proposed logic optimizations are only allowed to increase the bin area to the upper bound. Bounding the increase in bin area guarantees convergence of the placement process.

The present invention may be embodied in various forms, including computer-implemented methods, computer systems configured to implement such methods, computer-readable media containing instructions for implementing such methods, etc. Examples of computer-implemented methods embodying the invention have been described. Reducing such methods to tangible form as computerreadable media may be accomplished by methods well-known in the art.

Referring to FIG. 7, a diagram is shown of a computer system that may be used to practice the present invention. Attached to a system bus are one or more CPUs, read-only memory (ROM), read/write memory (RAM), mass storage, and other I/O devices. The other I/O devices will typically include a keyboard, a pointing device, and a display, and may further include any of a wide variety of commercially-available I/O devices, including, for example, magnetic storage devices, optical storage devices, other storage devices, printers, etc. Stored within memory (e.g., RAM) is software (e.g., EDA software) implementing methods of the type previously described.

New deep submicron technologies are resulting in a much stronger dependence between the steps of logic optimization, cell placement and interconnection routing. Consequently, current design methodologies that handle these steps separately result in too many iterations over these steps and possibly no convergence, causing long delays in the design process. This invention will significantly reduce, if not eliminate, the iterations needed by considering not only the impact of interconnect during logic optimization of area/timing, but also at the same time doing logic optimization to help placement relieve congestion and thus generate a circuit that is easily routable.

It will be appreciated by those of ordinary skill in the art that the invention can be embodied in other specific forms without departing from the spirit or essential character thereof The presently disclosed embodiments are therefore considered in all respects to be illustrative and not restrictive. The scope of the invention is indicated by the appended claims rather than the foregoing description, and all changes which come within the meaning and range of equivalents thereof are intended to be embraced therein.

What is claimed is:

1. A method of modifying an integrated circuit design to facilitate placement of circuit elements within one or more regions called bins on an integrated circuit design layout, comprising the steps of:

performing an initial placement of integrated circuit elements within bins on the design layout;

calculating congestion of the initial placement; and

subject to limits on the increase in area of integrated circuit elements within a bin, performing logic modifications within selected bins of the integrated circuit design to allow congestion of the placement to be improved.

2. The method of claim 1, comprising the further step of performing placement refinement in an attempt to improve congestion by taking advantage of the logic modifications.

3. The method of claim 2, comprising the further steps of:

tracking logic modifications to determine which logic modifications resulted in placement modifications during placement refinement; and

undoing logic modifications that did not result in placement modifications.

4. The method of claim 2, comprising the further step of modifying logic within the integrated circuit design to improve timing performance of the integrated circuit design subject to limits on the increase in area of integrated circuit elements within a bin.

5. The method of claim 4, wherein modifying logic to improve timing performance comprises speeding up part of the circuit to improve timing slack in that part of the circuit.

6. The method of claim 2, comprising the further steps of:

calculating congestion of the placement following placement refinement; and

depending on the degree to which congestion has been improved, repeating said steps of modifying logic and performing placement refinement.

7. The method of claim 2, wherein modifying logic comprises replacing an original set of gates in the circuit with a different set of gates that is logically equivalent to the original set of gates.

8. The method of claim 7, wherein the different set of gates results in a lower ratio of number of pins to routable area in at least one bin.

9. The method of claim 7, wherein modifying logic comprises replacing a single gate having a plural number N of fanouts with a plurality of gates each having fewer than N fanouts.

10. The method of claim 7, wherein modifying logic comprises inserting buffers within a fanout tree of a gate.

11. The method of claim 7, wherein modifying logic comprises replacing a single gate having a plural number N of fanins with a plurality of gates each having fewer than N fanins.

12. A method of modifying an integrated circuit design to facilitate placement of circuit elements within one or more regions called bins on an integrated circuit design layout, comprising the steps of:

performing an initial placement of integrated circuit elements within bins on the design layout, connections between the integrated circuit elements being represented as nets within a netlist describing the integrated circuit design;

calculating congestion of the initial placement; and

subject to limits on the increase in area of integrated circuit elements within a bin, performing logic modifications within selected bins of the integrated circuit design;

wherein the logic modifications improve timing of selected nets belonging to the selected bins, reducing constraints on a subsequent placement step.

13. The method of claim 12, comprising the further step of performing placement refinement in an attempt to improve congestion by taking advantage of the logic modifications.

14. The method of claim 13, comprising the further steps of:

tracking logic modifications to determine which logic modifications resulted in placement modifications during placement refinement; and

undoing logic modifications that did not result in placement modifications.

15. A computer-readable medium including instructions for modifying an integrated circuit design to facilitate placement of circuit elements within one or more regions called bins on an integrated circuit design layout, including instructions for:

performing an initial placement of integrated circuit elements within bins on the design layout;

calculating congestion of the initial placement; and

subject to limits on the increase in area of integrated circuit elements within a bin, performing logic modifications within selected bins of the integrated circuit design to allow congestion of the placement to be improved.

16. A computer-readable medium including instructions for modifying an integrated circuit design to facilitate placement of circuit elements within one or more regions called bins on an integrated circuit design layout, including instructions for:

performing an initial placement of integrated circuit elements within bins on the design layout, connections between the integrated circuit elements being represented as nets within a netlist describing the integrated circuit design;

calculating congestion of the initial placement; and

subject to limits on the increase in area of integrated circuit elements within a bin, performing logic modifications within selected bins of the integrated circuit design;

wherein the logic modifications improve timing of selected nets belonging to the selected bins, reducing constraints on a subsequent placement step.

17. Apparatus for modifying an integrated circuit design to facilitate placement of circuit elements within one or more regions called bins on an integrated circuit design layout, comprising:

means for performing an initial placement of integrated circuit elements within bins on the design layout;

means for calculating congestion of the initial placement; and

means for, subject to limits on the increase in area of integrated circuit elements within a bin, performing logic modifications within selected bins of the integrated circuit design to allow congestion of the placement to be improved.

18. Apparatus for modifying an integrated circuit design to facilitate placement of circuit elements within one or more regions called bins on an integrated circuit design layout, comprising:

means for performing an initial placement of integrated circuit elements within bins on the design layout, connections between the integrated circuit elements being represented as nets within a netlist describing the integrated circuit design;

means for calculating congestion of the initial placement; and

means, subject to limits on the increase in area of integrated circuit elements within a bin, performing logic modifications within selected bins of the integrated circuit design;

wherein the logic modifications improve timing of selected nets belonging to the selected bins, reducing constraints on a subsequent placement step.

* * * * *

TAB 2

IW 1491145

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office

August 02, 2006

THIS IS TO CERTIFY THAT ANNEXED IS A TRUE COPY FROM THE RECORDS OF THIS OFFICE OF THE FILE WRAPPER AND CONTENTS OF:

APPLICATION NUMBER: *09/097,076*
FILING DATE: *June 12, 1998*
PATENT NUMBER: *6,192,508*
ISSUE DATE: *February 20, 2001*

By Authority of the
Under Secretary of Commerce for Intellectual Property
and Director of the United States Patent and Trademark Office

H. L. Jackson

H. L. JACKSON
Certifying Officer

CERTIFICATE OF FACSIMILE TRANSMISSION

I HEREBY CERTIFY THAT THIS CORRESPONDENCE IS BEING FACSIMILE TRANSMITTED TO EXAMINER, HUGH JONES, ART UNIT 2763, U.S. PATENT AND TRADEMARK OFFICE ON THE DATE SHOWN BELOW.

Name of person signing certification: Sharon E. Byam

Date: March 7, 2000 Signature: Sharon E. Byam

Resubmitted April 18, 2000 Sharon E Byam

Patent

Attorney's Docket No. 032260-004

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of	)	**BOX AF**
	)	
Sharad Malik, et al.	)	Group Art Unit: 2763
	)	
Application No.: 09/097,076	)	Examiner: Jones, H.
	)	
Filed: June 12, 1998	)	
	)	
For: METHOD FOR LOGIC OPTIMIZATION FOR IMPROVING TIMING AND CONGESTION DURING PLACEMENT IN INTEGRATED CIRCUIT DESIGN	)	

AMENDMENT UNDER 37 C.F.R. §312

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

Please amend this application as follows:

IN THE CLAIMS:

1. A method of modifying an integrated circuit design to facilitate placement of circuit elements within one or more regions called bins on an integrated circuit design grid, comprising the steps of:

performing an initial placement of integrated circuit elements within bins on the design grid;

calculating congestion of the initial placement; and

a1

subject to limits on the increase in area of integrated circuit elements within a bin, performing logic modifications within selected bins of the integrated circuit design to allow congestion of the placement to be improved.

12. A method of modifying an integrated circuit design to facilitate placement of circuit elements within one or more regions called bins on an integrated circuit design grid, comprising the steps of:

a2

performing an initial placement of integrated circuit elements within bins on the design grid, connections between the integrated circuit elements being represented as nets within a netlist describing the integrated circuit design;

calculating congestion of the initial placement; and

subject to limits on the increase in area of integrated circuit elements within a bin, performing logic modifications within selected bins of the integrated circuit design to allow congestion of the placement to be improved;

wherein the logic modifications improve timing of selected nets belonging to the selected bins, reducing constraints on a subsequent placement step.

15. A computer-readable medium including instructions for modifying an integrated circuit design to facilitate placement of circuit elements within one or more regions called bins on an integrated circuit design grid, including instructions for:

a3

performing an initial placement of integrated circuit elements within bins on the design grid;

calculating congestion of the initial placement; and

subject to limits on the increase in area of integrated circuit elements within a bin, performing logic modifications within selected bins of the integrated circuit design to allow congestion of the placement to be improved.

16. A computer-readable medium including instructions for modifying an integrated circuit design to facilitate placement of circuit elements within one or more regions called bins on an integrated circuit design grid, including instructions for:

performing an initial placement of integrated circuit elements within bins on the design grid, connections between the integrated circuit elements being represented as nets within a netlist describing the integrated circuit design;

calculating congestion of the initial placement; and

subject to limits on the increase in area of integrated circuit elements within a bin, performing logic modifications within selected bins of the integrated circuit design to allow congestion of the placement to be improved;

wherein the logic modifications improve timing of selected nets belonging to the selected bins, reducing constraints on a subsequent placement step.

G3

17. Apparatus for modifying an integrated circuit design to facilitate placement of circuit elements within one or more regions called bins on an integrated circuit design grid, comprising:

means for performing an initial placement of integrated circuit elements within bins on the design grid;

means for calculating congestion of the initial placement; and

means for, subject to limits on the increase in area of integrated circuit elements within a bin, performing logic modifications within selected bins of the integrated circuit design to allow congestion of the placement to be improved.

18. Apparatus for modifying an integrated circuit design to facilitate placement of circuit elements within one or more regions called bins on an integrated circuit design grid, comprising:

means for performing an initial placement of integrated circuit elements within bins on the design grid, connections between the integrated circuit elements being represented as nets within a netlist describing the integrated circuit design;

a3

means for calculating congestion of the initial placement; and

means, subject to limits on the increase in area of integrated circuit elements within a bin, performing logic modifications within selected bins of the integrated circuit design to allow congestion of the placement to be improved;

wherein the logic modifications improve timing of selected nets belonging to the selected bins, reducing constraints on a subsequent placement step.

REMARKS

By the present amendment, the claims would be amended to account for the possibility of performing the present invention using only a single bin (i.e., one encompassing the entire integrated circuit) as opposed to multiple bins. This change is not believed to affect patentability of the claims. Entry of the amendment is respectfully requested.

Respectfully submitted,

BURNS, DOANE, SWECKER & MATHIS, L.L.P.

By: _____
Michael J. Ure
Registration No. 33,089

P.O. Box 1404
Alexandria, Virginia 22313-1404
(650) 622-2300

Date: March 6, 2000

TAB 3

IW 1491145

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office

August 02, 2006

THIS IS TO CERTIFY THAT ANNEXED IS A TRUE COPY FROM THE RECORDS OF THIS OFFICE OF THE FILE WRAPPER AND CONTENTS OF:

APPLICATION NUMBER: *09/097,076*
FILING DATE: *June 12, 1998*
PATENT NUMBER: *6,192,508*
ISSUE DATE: *February 20, 2001*

By Authority of the
Under Secretary of Commerce for Intellectual Property
and Director of the United States Patent and Trademark Office

H. L. Jackson

H. L. JACKSON
Certifying Officer

UNITED STATES DEPARTMENT OF COMMERCE
Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C 20231

APPLICATION NUMBER	FILING DATE	FIRST NAMED APPLICANT	ATTORNEY DOCKET NO
09/097,078	06/12/98	MALI	5

CM

LM02/0504

BURNS DOANE SWECKER & MATHIS
P O BOX 1404
ALEXANDRIA VA 22313 1404

EXAMINER
JONES,H

ART UNIT	PAPER NUMBER
1743	10

DATE MAILED: 05/02/00

Response to Rule 312 Communication

☐ The petition filed _____ under 37 CFR 1.312(b) is granted. The paper has been forwarded to the examiner for consideration on the merits.

_____

_____
Director,
Patent Examining Group _____

☑ The amendment filed 4/18/2000 under 37 CFR 1.312 has been considered, and has been:

☑ entered.

☐ entered as directed to matters of form not affecting the scope of the invention (Order 3311).

☐ disapproved. See explanation below.

☐ entered in part. See explanation below.

KEVIN J. TESKA
SUPERVISORY
PATENT EXAMINER

FORM PTOL-271 (REV 1/96)

★ U.S. GPO. 1996-404-498/40514

TAB 4

THE AMERICAN HERITAGE DICTIONARY OF THE ENGLISH LANGUAGE

DICT

THE AMERICAN HERITAGE
ONARY
OF THE ENGLISH LANGUAGE

WILLIAM MORRIS, Editor

Published by

HOUGHTON MIFFLIN COMPANY / BOSTON
ATLANTA / DALLAS / GENEVA, ILLINOIS / HOPEWELL, NEW JERSEY / PALO ALTO

Standard Book Numbers: 395–09064–4 (de luxe edition);
395–09066–0 (standard edition; thumb-indexed); 395–09070–9 (school edition);
0–395–20359–7 (new college edition; plain edges);
0–395–20360–0 (new college edition; thumb-indexed)

Library of Congress Catalog Card Number 76–86995

Manufactured in the United States of America

Computer-composed by Inforonics, Inc.
in Maynard, Massachusetts

refundere (past participle *refūsus*), to pour back. See **refund.**] —re-fus'er *n.*

Synonyms: *refuse, decline, reject, spurn, rebuff, ignore.* These verbs imply negation or denial in varying manner. *Refuse* is used of a positive, unyielding, sometimes brusque decision not to act, accept, or do something. *Decline* involves withholding consent but usually doing so courteously: *decline an invitation.* *Reject* applies to a blunt, even hostile, refusal to do something. *Spurn* is to reject scornfully, contemptuously. *Rebuff* pertains to the disdainful refusal to accept something offered. *Ignore* implies affront in not recognizing the presence or existence of a person or proposal.

ref·use[2] (rĕf'yōōs) *n.* Anything discarded or rejected as useless or worthless; trash; rubbish. —*adj.* Discarded or rejected as useless or worthless. [Middle English, something rejected, from Old French *refus*, refusal, from *refuser*, to REFUSE.]

ref·u·ta·tion (rĕf'yōō-tā'shən) *n.* Also **re·fu·tal** (rĭ-fyōōt'l). 1. The act of refuting. 2. Something that refutes.

re·fute (rĭ-fyōōt') *tr.v.* **-futed, -futing, -futes.** To prove (a statement or argument) to be false or erroneous; disprove. 2. To prove (a person) to be wrong. —See Synonyms at **deny.** [Latin *refūtāre*, rebut, drive back. See **bheu-** in Appendix.*] —re·fut'a·bil'i·ty (rĭ-fyōō'tə-bĭl'ə-tē, rĕf'yə-tə-) *n.* —re·fut'a·ble *adj.* —re·fut'a·bly *adv.* —re·fut'er *n.*

reg. 1. regent. 2. regiment. 3. region. 4. register; registered. 5. registrar. 6. registry. 7. regular; regularly. 8. regulation. 9. regulator.

re·gain (rē-gān') *tr.v.* **-gained, -gaining, -gains.** 1. To recover possession of; get back again. 2. To manage to reach again. —See Synonyms at **recover.** —re·gain'er *n.*

re·gal (rē'gəl) *adj.* 1. Of or pertaining to a king; royal. 2. Belonging to or befitting a king: *regal attire.* [Middle English, from Old French, from Latin *rēgālis*, royal, from *rēx* (stem *rēg-*), king. See **reg-**[1] in Appendix.*] —re'gal·ly *adv.*

re·gale (rĭ-gāl') *v.* **-galed, -galing, -gales.** —*tr.* 1. To delight or entertain; give pleasure to. 2. To entertain sumptuously with food and drink; provide a feast for. —*intr.* To feast. —See Synonyms at **amuse.** —*n. Obsolete.* 1. A great feast; a sumptuous repast. 2. A choice food or drink; delicacy. 3. Refreshment. [French *régaler*, from Old French *regaler*, from *regal*, REGAL.] —re·gale'ment *n.*

re·ga·lia (rĭ-gāl'yə, -gā'lē-ə) *n.* Plural in form, often used with a singular verb. 1. The emblems and symbols of royalty, as the crown and scepter. 2. The rights and privileges of royalty. 3. The distinguishing symbols of any rank, office, order, or society. 4. Magnificent or fancy attire; finery. [Medieval Latin *rēgālia*, plural of *rēgāle*, royal prerogative, from Latin, neuter of *rēgālis*, REGAL.]

re·gal·i·ty (rĭ-gāl'ə-tē) *n., pl.* **-ties.** 1. Royalty or sovereignty; kingship. 2. A country or area under the authority of a monarch; kingdom. 3. The rights or privileges of a king.

re·gard (rĭ-gärd') *v.* **-garded, -garding, -gards.** —*tr.* 1. To look at attentively; observe closely. 2. To look upon or consider in a particular way: *I regard him as a fool.* 3. To have great affection or admiration for: *She regards her father highly.* 4. To relate, concern, or refer to: *This item regards your question.* 5. To consider or take into account: *regard the fact that man is mortal.* 6. *Obsolete.* To take care of. —*intr.* 1. To look; to gaze. 2. To give heed; pay attention. —See Synonyms at **consider.** —*n.* 1. A look or gaze. 2. Careful thought or attention; concern; heed: *He gives little regard to his appearance.* 3. Respect, affection, or esteem: *He has won the regard of all.* 4. *Plural.* Sentiments of respect or affection; good wishes: *send one's regards.* 5. Reference or relation: *in regard to this case.* 6. A particular point or respect: *I agree in this regard.* 7. *Obsolete.* Appearance or aspect. [Middle English *regarden*, from Old French *regarder*, *reguarder*, to look at, regard : *re-*, back, back at + *guarder*, *garder*, to GUARD.]

Synonyms: *regard, esteem, admiration, approbation.* These terms refer in varying degree to the appreciation of the worth of a person or thing. *Regard* is the least forceful, a general term implying affection and recognition of worth; in the plural it is often used as a courteous close to a letter or message: *with best regards.* *Esteem* connotes measured, considered, and pleasurable appraisal of worth: *esteem for a distinguished scholar.* *Admiration* is undisguised pleasure, delight, and wonder in contemplating something unusual, beautiful, or skillful. *Approbation* is the approval or commendation of something after careful consideration of its worth.

Usage: *Regard* (noun), in denoting reference, is found principally in the combinations *in* (or *with*) *regard to* (but not *in, or with, regards to*). The terms *regarding* and *respecting* (used as prepositions) and the expression *as regards* are acceptable in the same sense, though sometimes disparaged by stylists. Both *regard* and *respect* have the sense of *particular: In some respects the books are alike.* But *respect* is much more appropriate to such constructions; 84 per cent of the Usage Panel term *in some regards* unacceptable in the preceding example. *Regard*, as a verb equivalent to *consider*, is normally used with *as*, not with the infinitive *to be: I regard it as an insult* (rather than *I regard it an insult* or *regard it to be an insult*).

re·gar·dant (rĭ-gär'dənt) *adj. Heraldry.* With the face turned backward in profile. [Middle English, from Old French, from *regarder*, to REGARD.]

re·gard·ful (rĭ-gärd'fəl) *adj.* 1. Showing regard; observant; heedful. Often used with *of*. 2. Showing deference; respectful; considerate. —re·gard'ful·ly *adv.* —re·gard'ful·ness *n.*

re·gard·ing (rĭ-gär'dĭng) *prep.* In reference to; with respect to; concerning. See Usage note at **regard.**

re·gard·less (rĭ-gärd'lĭs) *adj.* Heedless; unmindful. Often used with *of*. —*adv.* In spite of everything; anyway.

re·gard·less·ly (rĭ-gärd'lĭs-lē) *adv.* In a regardless manner. —re·gard'less·ness *n.*

re·gat·ta (rĭ-gă'tə, -gät'ə) *n.* A boat race or an organized series of boat races. [Italian (Venetian dialect) *regatta*, *regata*†, gondola race.]

regd. registered.

re·ge·late (rē'jə-lāt', rē'jə-lāt') *intr.v.* **-lated, -lating, -lates.** To undergo regelation. [Back-formation from REGELATION.]

re·ge·la·tion (rē'jə-lā'shən) *n.* 1. The fusion of two blocks of ice by pressure. 2. Successive melting under pressure and freezing when pressure is relaxed at the interface of two blocks of ice. [RE- + GELATION.]

re·gen·cy (rē'jən-sē) *n., pl.* **-cies.** 1. The office, area of jurisdiction, or government of a regent or regents. 2. A person or group selected to govern in place of a king or other ruler in case of minority, absence, incompetence, or sickness. 3. The period during which a regent governs, as in England (1811–20) or France (1715–23). —*adj.* Also **Re·gen·cy.** Of, relating to, or characteristic of the style, especially in furniture, prevalent during the regency (1811–20) of George, Prince of Wales.

re·gen·er·a·cy (rĭ-jĕn'ər-ə-sē) *n.* The state of being regenerated.

re·gen·er·ate (rĭ-jĕn'ə-rāt') *v.* **-ated, -ating, -ates.** —*tr.* 1. To reform spiritually or morally. 2. To form, construct, or create anew. 3. *Biology.* To replace (a lost or damaged organ or part) by formation of new tissue. —*intr.* 1. To become formed or constructed again. 2. To undergo spiritual conversion or rebirth. 3. To effect regeneration. —*adj.* (rĭ-jĕn'ər-ĭt). 1. Spiritually or morally revitalized. 2. Restored; refreshed; renewed. [Latin *regenerāre*, to reproduce : *re-*, again + *generāre*, to beget, GENERATE.] —re·gen'er·a·tive (-ə-rā'tĭv, -ər-ə-tĭv) *adj.* —re·gen'er·a·tive·ly *adv.*

re·gen·er·a·tion (rĭ-jĕn'ə-rā'shən) *n.* 1. The act or process of regenerating or the state of being regenerated. 2. Spiritual or moral revival or rebirth. 3. *Biology.* The regrowth of lost or destroyed parts or organs.

re·gen·er·a·tor (rĭ-jĕn'ə-rā'tər) *n.* One that regenerates.

Re·gens·burg (rā'gəns-bōōrk'). Also **Ratisbon** (răt'ĭs-bŏn'). A city on the Danube in eastern Bavaria, West Germany. Population, 125,000.

re·gent (rē'jənt) *n. Abbr.* **reg., Regt.** 1. One who rules during the absence or disability of a sovereign. 2. One acting as a ruler or governor. 3. A person serving on a board that governs a university or other educational institution or system in the United States. [Middle English, from Old French, ruling, from Medieval Latin *regēns*, from Latin, present participle of *regere*, to rule. See **reg-**[1] in Appendix.*] —re'gent *adj.*

Reg·gio di Ca·la·bri·a (rād'jō dē kä-lä'brē-ä). Also **Reggio, Reggio Ca·la·bri·a.** A city in southern Italy, on the Strait of Messina opposite Sicily. Population, 157,000.

Reg·gio nell'E·mi·lia (rād'jō nāl'lä-mē'lyä). Also **Reggio, Reggio E·mi·lia** (rād'jō ā-mē'lyä). A city and manufacturing center of north-central Emilia-Romagna, Italy. Population, 122,000.

reg·i·cide (rĕj'ə-sīd') *n.* 1. The killing of a king. 2. One who kills or helps to kill a king. [Latin *rēx* (stem *rēg-*), king (see **reg-**[1] in Appendix*) + -CIDE.] —reg'i·cid'al (-sīd'l) *adj.*

re·gime (rā-zhēm', rĭ-) *n.* Also **ré·gime.** 1. A system of management of government; an administration. 2. A social system or pattern. 3. A regimen. [French *régime*, from Latin *regimen*, from *regere*, to rule. See **reg-**[1] in Appendix.*]

reg·i·men (rĕj'ə-mən, -mĕn') *n.* 1. Governmental rule or control. 2. The systematic procedure of a natural phenomenon or process. 3. A system of therapy: *a dietetic regimen.* [Middle English, from Latin, REGIME.]

reg·i·ment (rĕj'ə-mənt) *n. Abbr.* **reg., regt.** A military unit of ground troops, consisting of at least two battalions and sometimes other units. —*tr.v.* (rĕj'ə-mĕnt') **regimented, -menting, -ments.** 1. To organize or form into a regiment or regiments. 2. To appoint to a regiment. 3. To put into order; systematize. 4. To force uniformity and discipline upon. [Middle English, from Old French, from Late Latin *regimentum*, from Latin *regere*, to rule. See **reg-**[1] in Appendix.*] —reg'i·men'tal *adj.* —reg'i·men·ta'tion *n.*

reg·i·men·tals (rĕj'ə-mĕnt'əlz) *pl.n.* 1. The uniform and insignia characteristic of a particular regiment. 2. Military dress.

Re·gi·na (rĭ-jī'nə). The capital of Saskatchewan, Canada. Population, 131,000.

Reg·i·nald (rĕj'ə-nəld). A masculine given name. [Middle English *Reginaldus*, Old English *Regenweald* : *regen*, power (see **rēk-** in Appendix*) + *weald*, force (see **wal-** in Appendix*).]

Re·gi·o·mon·ta·nus (rē'jē-ō-mŏn-tā'nəs, -tă'nəs, -tăn'əs, rĕj'ē-). Original name, Johann Müller. 1436–1476. German mathematician and astronomer.

re·gion (rē'jən) *n. Abbr.* **reg.** 1. Any large, usually continuous segment of a surface or space; an area. 2. A large and indefinite portion of the earth's surface. 3. A specified district or territory. 4. A field of interest or activity; sphere. 5. A part of the earth characterized by distinctive animal or plant life. 6. An area of the body having natural or arbitrarily assigned boundaries: *the abdominal region.* —See Synonyms at **area.** [Middle English *regioun*, kingdom, from Old French *region*, from Latin *regiō*, direction, boundary, from *regere*, to direct. See **reg-**[1] in Appendix.*]

re·gion·al (rē'jən-əl) *adj.* 1. Of, pertaining to, or characteristic of a large geographic region. 2. Of, pertaining to, or characteristic of a particular region or district; localized. 3. Of, belonging to, or characteristic of a form of a language that is

TAB 5

THE AMERICAN HERITAGE DICTIONARY OF THE ENGLISH LANGUAGE

DICT

THE AMERICAN HERITAGE

ONARY

OF THE ENGLISH LANGUAGE

WILLIAM MORRIS, Editor

Published by

HOUGHTON MIFFLIN COMPANY / BOSTON
ATLANTA / DALLAS / GENEVA, ILLINOIS / HOPEWELL, NEW JERSEY / PALO ALTO

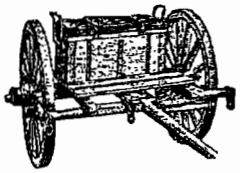
All correspondence and inquiries should be directed to
Dictionary Division, Houghton Mifflin Company
One Beacon Street, Boston, Massachusetts 02107

Standard Book Numbers: 395–09064–4 (de luxe edition);
395–09066–0 (standard edition; thumb-indexed); 395–09070–9 (school edition);
0–395–20359–7 (new college edition; plain edges);
0–395–20360–0 (new college edition; thumb-indexed)

Library of Congress Catalog Card Number 76–86995

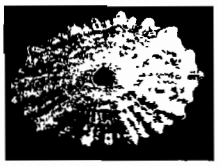
Manufactured in the United States of America

Computer-composed by Inforonics, Inc.
in Maynard, Massachusetts



limber²

lime¹



limpet
Fissurella barbadensis

member, or representative of a larger body, group, or the like. **5.** *Informal.* An impish or naughty child. —**out on a limb.** *Informal.* In a difficult, awkward, or vulnerable position. —*tr.v.* **limbed, limbing, limbs.** To dismember. [Middle English *lim, lymm,* Old English *lim,* akin to Old Norse *limr*†.]

limb² (lĭm) *n.* **1.** *Astronomy.* The circumferential edge of the apparent disk of a celestial body. **2.** The edge of a graduated arc or circle used in an instrument to measure angles. **3.** *Botany.* The expanded tip of a petal or the expanded upper part of a united corolla. [French *limbe,* from Latin *limbus,* border, hem, seam. See **limbus.**]

lim·bate (lĭm′bāt′) *adj. Botany.* Having an edge or margin of a different color. [Late Latin *limbātus,* bordered, from *limbus,* border. See **limbus.**]

lim·ber¹ (lĭm′bər) *adj.* **1.** Bending or flexing readily; pliable. **2.** Capable of moving, bending, or contorting easily; agile; supple. —*v.* **limbered, -bering, -bers.** —*tr.* To make limber. Often used with *up.* —*intr.* To make oneself limber. Used with *up: The football players limbered up before the game.* [Origin uncertain.] —**lim′ber·ly** *adv.* —**lim′ber·ness** *n.*

lim·ber² (lĭm′bər) *n.* A two-wheeled horse-drawn vehicle that carries ammunition and behind which a field gun may be towed. —*v.* **limbered, -bering, -bers.** —*tr.* To fasten a limber to (a gun). Often used with *up.* —*intr.* To fasten a limber and a gun together. Often used with *up.* [Middle English *lymo(u)r,* shaft of a carriage, possibly from Medieval Latin *limōnārius,* of a shaft, from *limō,* shaft, perhaps from Celtic.]

lim·bo (lĭm′bō) *n., pl.* **-bos. 1.** *Often capital* L. *Theology.* The abode of souls kept from Heaven through circumstance, such as lack of baptism. **2.** A region or condition of oblivion or neglect. **3.** A state or place of confinement. **4.** An intermediate place or state. [Middle English, from Medieval Latin *in limbō,* "(region) on the border (of hell)," : *in,* on + *limbus,* border (see **limbus.**)]

Lim·burg (lĭm′bûrg′). **1.** Also **Lim·bourg** (-bo͝org′). A province occupying 930 square miles in eastern Belgium. Population, 615,000. Capital, Hasselt. **2.** A province of the Netherlands, occupying 840 square miles in the southeast. Population, 954,000.

Lim·burg·er cheese (lĭm′bûr′gər). A soft white cheese with a very strong odor and flavor, originally produced in Limburg, Belgium. Also called "Limburger."

lim·bus (lĭm′bəs) *n., pl.* **-bi** (-bī′). *Biology.* A distinctive border or edge. [Latin *limbus*†, border, hem, seam.]

lime¹ (līm) *n.* **1.** A spiny tree, *Citrus aurantifolia,* native to Asia, having evergreen leaves, fragrant white flowers, and edible, egg-shaped fruit with a green rind and acid juice used as flavoring. **2.** The fruit of this tree. [French *lime,* from Provençal *limo,* from Arabic *limah.*]

lime² (līm) *n.* Any of several Old World linden trees. [Variant of *line,* dialectal variant of obsolete *lind,* LINDEN.]

lime³ (līm) *n.* **1. a.** *Chemistry.* **Calcium oxide** *(see).* **b.** Any of various mineral and industrial forms of calcium oxide differing chiefly in water content and percentage of such constituents as silica, alumina, and iron. **2.** A sticky substance smeared on twigs and used to catch small birds; birdlime. —*tr.v.* **limed, liming, limes. 1.** To treat with lime. **2.** To smear with birdlime. **3.** To catch or snare with or as with birdlime. [Middle English *lim,* Old English *līm.* See **lei-** in Appendix.*]

lime·ade (līm′ād′) *n.* A sweetened beverage of lime juice and plain or carbonated water.

lime·kiln (līm′kĭl′, -kĭln′) *n.* A furnace used to reduce naturally occurring forms of calcium carbonate to lime.

lime·light (līm′līt′) *n.* **1.** A stage light in which lime is heated to incandescence producing brilliant illumination. **2.** The brilliant light so produced. Also called "calcium light." **3.** A focus of public attention or notoriety. Preceded by *the.*

li·men (lī′mən) *n., pl.* **-mens** or **limina** (lĭm′ə-nə). The threshold of a physiological or psychological response. [From Latin *līmen,* threshold, akin to *limes,* boundary, LIMIT.] —**lim′i·nal** (lĭm′ə-nəl) *adj.*

lim·er·ick (lĭm′ər-ĭk) *n.* A light humorous or nonsensical verse of five anapestic lines usually with the rhyme scheme *aabba.* [From the line "Will you come up to Limerick?" (the refrain of a convivial verse in a similar form).]

Lim·er·ick (lĭm′ər-ĭk). A county of the Republic of Ireland, occupying 1,037 square miles in the southwest. Population, 133,000.

li·mes (lī′mēz′) *n., pl.* **limites** (lĭm′ə-tēz′). A fortified boundary, as of the Roman Empire. [Latin *limes,* boundary, LIMIT.]

lime·stone (līm′stōn′) *n.* A shaly or sandy sedimentary rock, chiefly $CaCO_3$, containing variable quantities of magnesium carbonate and quartz, used as a building stone, and in the manufacture of lime, carbon dioxide, and cement.

lime·twig (līm′twĭg′) *n.* **1.** A twig covered with birdlime to catch birds. **2.** A snare.

lime·wa·ter (līm′wô′tər, -wŏt′ər) *n.* A clear colorless alkaline aqueous solution of calcium hydroxide, used in calamine lotion and other skin preparations and sometimes as an antacid.

lim·ey (lī′mē) *n., pl.* **-eys.** *Slang.* **1.** A British seaman. **2.** An Englishman.

li·mic·o·line (lī-mĭk′ə-lĭn′, -lĭn) *adj.* Of or pertaining to shore birds, such as sandpipers, of the suborder Charadrii. [From New Latin *Limicolae* (former order name), "mud dwellers" : Latin *limus,* slime, mud (see **lei-** in Appendix*) + *-colae,* from *-colus,* -COLOUS.]

lim·it (lĭm′ĭt) *n.* *Abbr.* **lim. 1.** The point, edge, or line beyond which something cannot or may not proceed; the final or furthest confines, bounds, or restriction of something. **2.** *Usually plural.* The boundary surrounding a specific area; bounds: *within the city limits.* **3.** The greatest amount or number allowed. **4.** In games of chance, the largest amount which may be bet at one time. **5.** *Obsolete.* A region or section enclosed within or as if within boundaries. **6.** *Mathematics.* A number k that is approached by a function $f(x)$ as x approaches a if, for every positive number ϵ, there exists a number δ such that $|f(x)-k|<\epsilon$ if $0<|x-a|<\delta$. —See Synonyms at **boundary.** —**off limits.** Prohibited, especially to military personnel not on official business. —**the limit.** *Informal.* One that approaches or exceeds certain limits, as of credibility, forbearance, or acceptability. —*tr.v.* **limited, -iting, -its.** To confine or restrict within a limit or limits. [Middle English *limite,* from Latin *limes*† (stem *limit-*), borderline between fields, boundary.] —**lim′it·a·ble** *adj.* —**lim′i·ta′tive** *adj.* —**lim′it·er** *n.*

Synonyms: *limit, restrict, confine, circumscribe, bound.* These verbs mean to keep or contain within a specified area. *Limit* refers principally to establishing a maximum, as in quantity, degree, space, or time, beyond which a person or thing cannot or may not go. It is sometimes interchangeable with *restrict* and *confine,* but *restrict* and *confine* more often refer to keeping persons, things, or activities within a prescribed area: *messages limited to 50 words; a sale limited* (or *confined*) *to two days; a soldier restricted* (or *confined*) *to his quarters; wiretapping restricted* (or *confined*) *to cases involving national security. Circumscribe* is applied to encircling in a literal sense or, more often, to keeping something intangible, such as power, law, or influence, within specified and often narrow limits. *Bound* refers largely to setting geographical limits.

lim·i·tar·y (lĭm′ə-tĕr′ē) *adj. Archaic.* **1. a.** Of or relating to a limit or boundary. **b.** Limiting; restrictive. **2.** Limited.

lim·i·ta·tion (lĭm′ə-tā′shən) *n.* **1.** The act of limiting or the state of being limited. **2.** A restriction. **3.** *Law.* A limited period during which, by statute, an action may be brought.

lim·it·ed (lĭm′ə-tĭd) *adj. Abbr.* **Ld., Ltd., ltd. 1. a.** Having a limit or limits. **b.** Confined or restricted. **2.** Not attaining the highest goals or achievement: *a limited success.* **3.** Having governmental or ruling powers restricted by enforceable limitations, as a constitution or legislative body. **4.** *Chiefly British.* Limiting the liability of each stockholder or partner in a business to his actual investment: *a limited company.* **5.** Designating transportation facilities, such as trains or buses that make few stops and carry relatively few passengers. —*n. Abbr.* **Ltd., ltd., Ld.** A limited train or bus. —**lim′i·ted·ly** *adv.* —**lim′i·ted·ness** *n.*

limited edition. An edition, as of a book or set of books, limited to a specified number of copies.

limited monarchy. A **constitutional monarchy** *(see).*

li·mites. Plural of **limes.**

lim·it·less (lĭm′ĭt-lĭs) *adj.* **1.** Having no limit or limits. **2.** Unconfined or unrestricted. —See Synonyms at **infinite.**

limn (lĭm) *tr.v.* **limned, limning, limns. 1.** *Archaic.* To describe. **2.** *Archaic.* To depict by painting or drawing. **3.** *Obsolete.* To illuminate with paintings or drawing. [Middle English *limnen,* to illuminate (manuscript), shortened from *luminen,* from Old French *luminer,* from Latin *lūmināre,* from *lūmen,* light. See **leuk-** in Appendix.*] —**lim′ner** (lĭm′nər) *n.*

lim·net·ic (lĭm-nĕt′ĭk) *adj.* Of or occurring in the deeper, open waters of lakes or ponds. [From Greek *limnē*†, pool, lake.]

lim·nol·o·gy (lĭm-nŏl′ə-jē) *n.* The scientific study of the life and phenomena of lakes, ponds, and streams. [Greek *limnē,* pool, lake (see **limnetic**) + -LOGY.] —**lim′no·log′i·cal** (-nə-lŏj′ĭ-kəl) *adj.* —**lim′no·log′i·cal·ly** *adv.* —**lim·nol′o·gist** *n.*

Lim·nos. See **Lemnos.**

Li·moges¹ (lĭ-mōzh′; *French* lē-mōzh′) *n.* A variety of fine porcelain made at Limoges. Also called "Limoges ware."

Li·moges² (lĭ-mōzh′; *French* lē-mōzh′). An industrial city of west-central France. Population, 118,000.

lim·o·nene (lĭm′ə-nēn′) *n.* A liquid, $C_{10}H_{16}$, with a characteristic lemonlike fragrance, used as a solvent, wetting agent, and dispersing agent, and in the manufacture of resins. [French *limon,* lime, from Old French, LEMON + -ENE.]

li·mo·nite (lī′mə-nīt′) *n.* A widely occurring yellowish-brown to black natural iron oxide, essentially $FeO(OH)\cdot nH_2O$, used as an ore of iron. [German *Limonit,* "meadow ore," bog iron ore : Greek *leimōn*†, meadow + -ITE.] —**lī′mo·nit′ic** (-nĭt′ĭk) *adj.*

Li·mou·sin (lē-mōō-zăN′). A region and former province of west-central France.

lim·ou·sine (lĭm′ə-zēn′, lĭm′ə-zēn′) *n.* Any of various large passenger vehicles; especially, an automobile with an enclosed passenger compartment and an open but roofed driver's seat. [Originally a kind of flowing mantle or coat, popularized in LIMOUSIN.]

limp (lĭmp) *intr.v.* **limped, limping, limps. 1.** To walk lamely, especially with irregularity, as if favoring one leg. **2.** To move or proceed haltingly or unsteadily. —*n.* An irregular, jerky, or awkward gait. —*adj.* **limper, limpest. 1.** Lacking or having lost rigidity; flaccid; flabby. **2.** Lacking strength or firmness of character; weak. —**limp′ly** *adv.* —**limp′ness** *n.* [Probably shortened from obsolete *limphalt,* lame, ultimately from Old English *lemphealt, læmpihalt.* See **leb-¹** in Appendix.*]

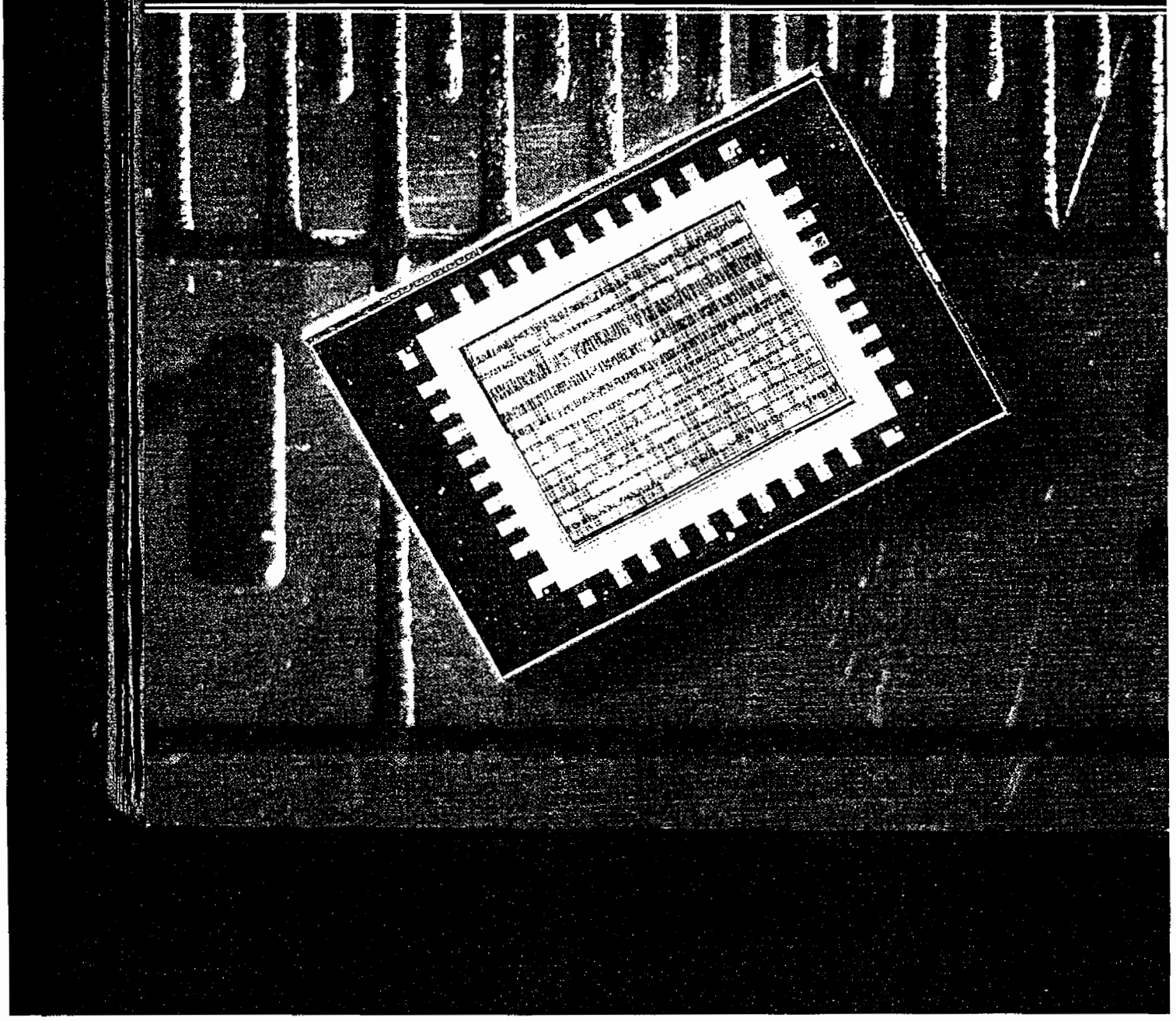
lim·pet (lĭm′pĭt) *n.* **1.** Any of numerous marine gastropod mollusks, as of the families Acmaeidae and Patellidae, characteristically having a tent-shaped shell and adhering to rocks of tidal areas. **2.** One who clings persistently. **3.** A type of explosive designed to cling to the hull of a ship and detonate on contact or signal. [Middle English *lempet,* Old English *lempedu,* from Medieval Latin *lamprēda,* LAMPREY.]

lim·pid (lĭm′pĭd) *adj.* **1.** Characterized by transparent clearness; pellucid. **2.** Easily intelligible; clear. **3.** Calm and untroubled;

TAB 6

Application-Specific Integrated Circuits

Michael John Sebastian Smith



Application-Specific Integrated Circuits

Michael John Sebastian Smith
University of Hawaii
Compass Design Automation

ADDISON-WESLEY

An imprint of Addison Wesley Longman, Inc.

Reading, Massachusetts • Harlow, England • Menlo Park, California • Berkeley, California
Don Mills, Ontario • Sydney • Bonn • Amsterdam • Tokyo • Mexico City

This book is in the Addison-Wesley VLSI Systems Series
Lynn Conway and Charles Seitz, *Consulting Editors*

Sponsoring Editor	Peter Gordon
Associate Editor	Helen Goldstein
Senior Production Supervisor	Juliet Silveri
Copyeditor/Proofreader	Cynthia Benn
Cover Design Supervisor	Simone Payment
Marketing Manager	Tracy Russ
Manufacturing Manager	Roy Logan

Library of Congress Cataloging-in-Publication Data

Smith, Michael J. S. (Michael John Sebastian)
Application-specific integrated circuits / Michael J.S. Smith.
p. cm.
Includes bibliographical references and index.
ISBN 0-201-50022-1
1. Application-specific integrated circuits. I. Title.
TK7874.6.S63 1997
621.39'5--dc20 93-32538
CIP

1 2 3 4 5 6 7 8 9 10-CRW-00999897

Designing a clock tree that balances the rise and fall times at the leaf nodes has the beneficial side-effect of minimizing the effect of **hot-electron wearout**. This problem occurs when an electron gains enough energy to become "hot" and jump out of the channel into the gate oxide (the problem is worse for electrons in *n*-channel devices because electrons are more mobile than holes). The trapped electrons change the threshold voltage of the device and this alters the delay of the buffers. As the buffer delays change with time, this introduces unpredictable skew. The problem is worst when the *n*-channel device is carrying maximum current with a high voltage across the channel—this occurs during the rise-and fall-time transitions. Balancing the rise and fall times in each buffer means that they all wear out at the same rate, minimizing any additional skew.

A **phase-locked loop** (PLL) is an electronic flywheel that locks in frequency to an input clock signal. The input and output frequencies may differ in phase, however. This means that we can, for example, drive a clock network with a PLL in such a way that the output of the clock network is locked in phase to the incoming clock, thus eliminating the latency of the clock network. A PLL can also help to reduce random variation of the input clock frequency, known as **jitter**, which, since it is unpredictable, must also be discounted from the time available for computation in each clock cycle. Actel was one of the first FPGA vendors to incorporate PLLs, and Actel's online product literature explains their use in ASIC design.

16.2 Placement

After completing a floorplan we can begin placement of the logic cells within the flexible blocks. Placement is much more suited to automation than floorplanning. Thus we shall need measurement techniques and algorithms. After we complete floorplanning and placement, we can predict both intrablock and interblock capacitances. This allows us to return to logic synthesis with more accurate estimates of the capacitive loads that each logic cell must drive.

16.2.1 Placement Terms and Definitions

CBIC, MGA, and FPGA architectures all have rows of logic cells separated by the interconnect—these are **row-based ASICs**. Figure 16.18 shows an example of the interconnect structure for a CBIC. Interconnect runs in horizontal and vertical directions in the channels and in the vertical direction by crossing through the logic cells. Figure 16.18(c) illustrates the fact that it is possible to use **over-the-cell routing** (**OTC** routing) in areas that are not blocked. However, OTC routing is complicated by the fact that the logic cells themselves may contain metal on the routing layers. We shall return to this topic in Section 17.2.7, "Multilevel Routing." Figure 16.19 shows the interconnect structure of a two-level metal MGA.

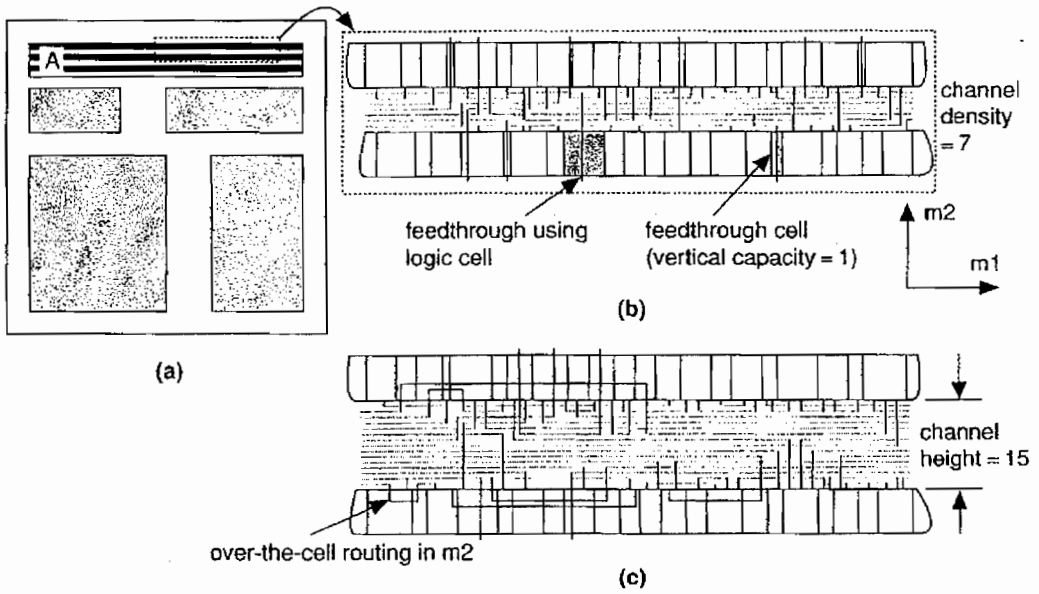


FIGURE 16.18 Interconnect structure. (a) The two-level metal CBIC floorplan shown in Figure 16.11b. (b) A channel from the flexible block A. This channel has a channel height equal to the maximum channel density of 7 (there is room for seven interconnects to run horizontally in m1). (c) A channel that uses OTC (over-the-cell) routing in m2.

Most ASICs currently use two or three levels of metal for signal routing. With two layers of metal, we route within the rectangular channels using the first metal layer for horizontal routing, parallel to the channel spine, and the second metal layer for the vertical direction (if there is a third metal layer it will normally run in the horizontal direction again). The maximum number of horizontal interconnects that can be placed side by side, parallel to the channel spine, is the **channel capacity**.

Vertical interconnect uses **feedthroughs** (or **feedthrus** in the United States) to cross the logic cells. Here are some commonly used terms with explanations (there are no generally accepted definitions):

- An unused **vertical track** (or just **track**) in a logic cell is called an **uncommitted feedthrough** (also **built-in feedthrough, implicit feedthrough**, or **jumper**).
- A vertical strip of metal that runs from the top to bottom of a cell (for **double-entry cells**), but has no connections inside the cell, is also called a feedthrough or jumper.

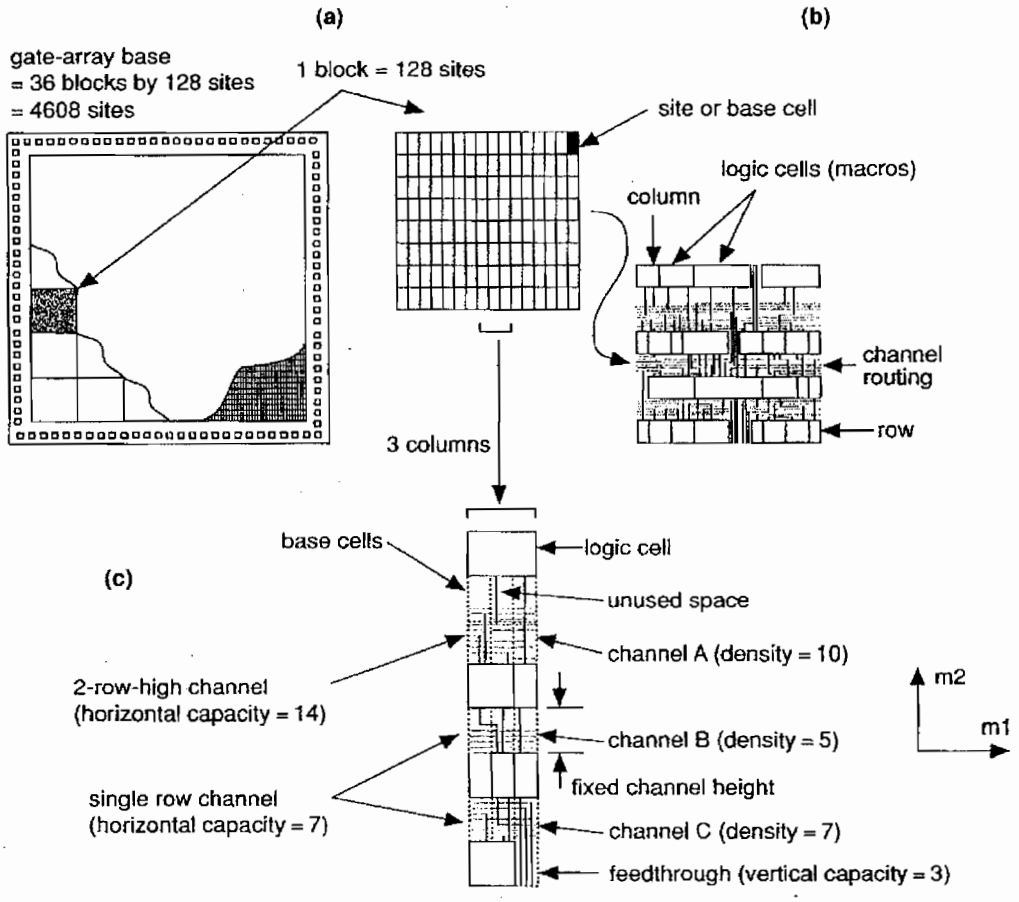


FIGURE 16.19 Gate-array interconnect. (a) A small two-level metal gate array (about 4.6 k-gate). (b) Routing in a block. (c) Channel routing showing channel density and channel capacity. The channel height on a gate array may only be increased in increments of a row. If the interconnect does not use up all of the channel, the rest of the space is wasted. The interconnect in the channel runs in m1 in the horizontal direction with m2 in the vertical direction.

- Two connectors for the same physical net are **electrically equivalent connectors** (or **equipotential connectors**). For double-entry cells these are usually at the top and bottom of the logic cell.
- A dedicated **feedthrough cell** (or **crosser cell**) is an empty cell (with no logic) that can hold one or more vertical interconnects. These are used if there are no other feedthroughs available.

- A **feedthrough pin** or **feedthrough terminal** is an input or output that has connections at both the top and bottom of the standard cell.
- A **spacer cell** (usually the same as a feedthrough cell) is used to fill space in rows so that the ends of all rows in a flexible block may be aligned to connect to power buses, for example.

There is no standard terminology for connectors and the terms can be very confusing. There is a difference between connectors that are joined inside the logic cell using a high-resistance material such as polysilicon and connectors that are joined by low-resistance metal. The high-resistance kind are really two separate **alternative connectors** (that cannot be used as a feedthrough), whereas the low-resistance kind are electrically equivalent connectors. There may be two or more connectors to a logic cell, which are not joined inside the cell, and which must be joined by the router (**must-join connectors**).

There are also **logically equivalent connectors** (or functionally equivalent connectors, sometimes also called just equivalent connectors—which is very confusing). The two inputs of a two-input NAND gate may be logically equivalent connectors. The placement tool can swap these without altering the logic (but the two inputs may have different delay properties, so it is not always a good idea to swap them). There can also be **logically equivalent connector groups**. For example, in an OAI22 (OR-AND-INVERT) gate there are four inputs: A1, A2 are inputs to one OR gate (gate A), and B1, B2 are inputs to the second OR gate (gate B). Then group A = (A1, A2) is logically equivalent to group B = (B1, B2)—if we swap one input (A1 or A2) from gate A to gate B, we must swap the other input in the group (A2 or A1).

In the case of channeled gate arrays and FPGAs, the horizontal interconnect areas—the channels, usually on m1—have a fixed capacity (sometimes they are called **fixed-resource ASICs** for this reason). The channel capacity of CBICs and channelless MGAs can be expanded to hold as many interconnects as are needed. Normally we choose, as an objective, to minimize the number of interconnects that use each channel. In the vertical interconnect direction, usually m2, FPGAs still have fixed resources. In contrast the placement tool can always add vertical feedthroughs to a channeled MGA, channelless MGA, or CBIC. These problems become less important as we move to three and more levels of interconnect.

16.2.2 Placement Goals and Objectives

The goal of a placement tool is to arrange all the logic cells within the flexible blocks on a chip. Ideally, the objectives of the placement step are to

- Guarantee the router can complete the routing step
- Minimize all the critical net delays
- Make the chip as dense as possible

We may also have the following additional objectives:

- Minimize power dissipation
- Minimize cross talk between signals

Objectives such as these are difficult to define in a way that can be solved with an algorithm and even harder to actually meet. Current placement tools use more specific and achievable criteria. The most commonly used placement objectives are one or more of the following:

- Minimize the total estimated interconnect length
- Meet the timing requirements for critical nets
- Minimize the interconnect congestion

Each of these objectives in some way represents a compromise.

16.2.3 Measurement of Placement Goals and Objectives

In order to determine the quality of a placement, we need to be able to measure it. We need an approximate measure of interconnect length, closely correlated with the final interconnect length, that is easy to calculate.

The graph structures that correspond to making all the connections for a net are known as **trees on graphs** (or just **trees**). Special classes of trees—**Steiner trees**—minimize the total length of interconnect and they are central to ASIC routing algorithms. Figure 16.20 shows a minimum Steiner tree. This type of tree uses diagonal connections—we want to solve a restricted version of this problem, using interconnects on a rectangular grid. This is called **rectilinear routing** or **Manhattan routing** (because of the east–west and north–south grid of streets in Manhattan). We say that the **Euclidean distance** between two points is the straight-line distance ("as the crow flies"). The **Manhattan distance** (or rectangular distance) between two points is the distance we would have to walk in New York.

The **minimum rectilinear Steiner tree** (MRST) is the shortest interconnect using a rectangular grid. The determination of the MRST is in general an NP-complete problem—which means it is hard to solve. For small numbers of terminals heuristic algorithms do exist, but they are expensive to compute. Fortunately we only need to estimate the length of the interconnect. Two approximations to the MRST are shown in Figure 16.21.

The **complete graph** has connections from each terminal to every other terminal [Hanan, Wolff, and Agule, 1973]. The **complete-graph measure** adds all the interconnect lengths of the complete-graph connection together and then divides by $n/2$, where n is the number of terminals. We can justify this since, in a graph with n terminals, $(n-1)$ interconnects will emanate from each terminal to join the other $(n-1)$ terminals in a complete graph connection. That makes $n(n-1)$ interconnects in total. However, we have then made each connection twice. So there are one-half

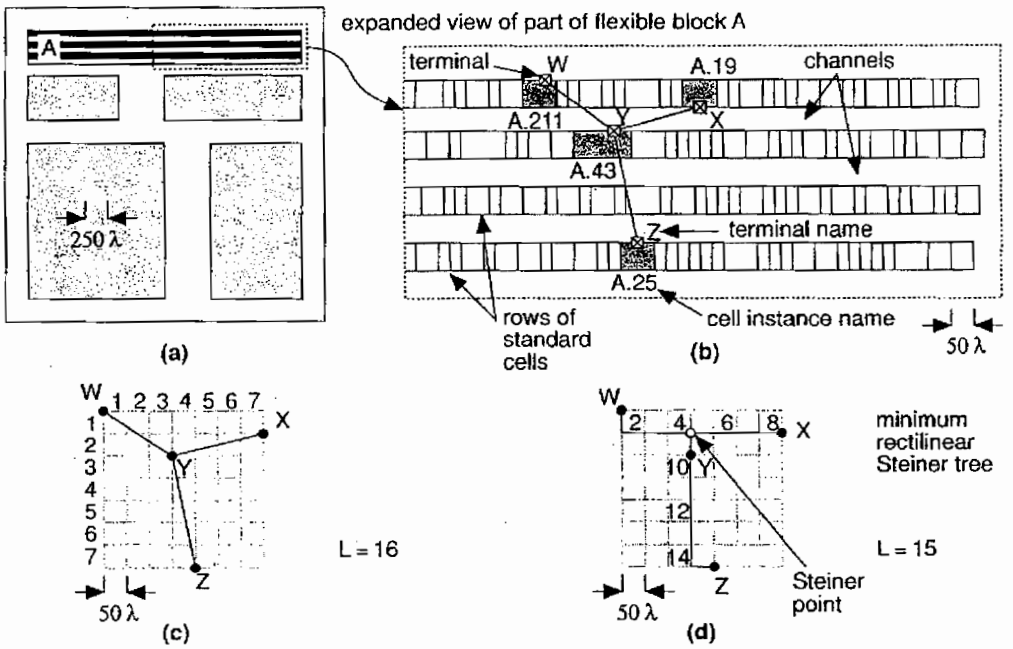


FIGURE 16.20 Placement using trees on graphs. (a) The floorplan from Figure 16.11b. (b) An expanded view of the flexible block A showing four rows of standard cells for placement (typical blocks may contain thousands or tens of thousands of logic cells). We want to find the length of the net shown with four terminals, W through Z, given the placement of four logic cells (labeled: A.211, A.19, A.43, A.25). (c) The problem for net (W, X, Y, Z) drawn as a graph. The shortest connection is the minimum Steiner tree. (d) The minimum rectilinear Steiner tree using Manhattan routing. The rectangular (Manhattan) interconnect-length measures are shown for each tree.

this many, or $n(n-1)/2$, interconnects needed for a complete graph connection. Now we actually only need $(n-1)$ interconnects to join n terminals, so we have $n/2$ times as many interconnects as we really need. Hence we divide the total net length of the complete graph connection by $n/2$ to obtain a more reasonable estimate of minimum interconnect length. Figure 16.21(a) shows an example of the complete-graph measure.

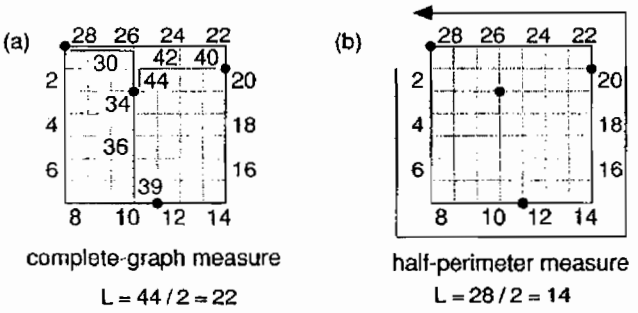


FIGURE 16.21 Interconnect-length measures. (a) Complete-graph measure. (b) Half-perimeter measure.

The **bounding box** is the smallest rectangle that encloses all the terminals (not to be confused with a logic cell bounding box, which encloses all the layout in a logic cell). The **half-perimeter measure** (or bounding-box measure) is one-half the perimeter of the bounding box (Figure 16.21b) [Schweikert, 1976]. For nets with two or three terminals (corresponding to a fanout of one or two, which usually includes over 50 percent of all nets on a chip), the half-perimeter measure is the same as the minimum Steiner tree. For nets with four or five terminals, the minimum Steiner tree is between one and two times the half-perimeter measure [Hanan, 1966]. For a circuit with m nets, using the half-perimeter measure corresponds to minimizing the cost function,

$$f = \frac{1}{2}\sum_{i=1}^{m} h_i \,, \tag{16.5}$$

where h_i is the half-perimeter measure for net i.

It does not really matter if our approximations are inaccurate if there is a good correlation between actual interconnect lengths (after routing) and our approximations. Figure 16.22 shows that we can adjust the complete-graph and half-perimeter measures using correction factors [Goto and Matsuda, 1986]. Now our wiring length approximations are functions, not just of the terminal positions, but also of the number of terminals, and the size of the bounding box. One practical example adjusts a Steiner-tree approximation using the number of terminals [Chao, Nequist, and Vuong, 1990]. This technique is used in the Cadence Gate Ensemble placement tool, for example.

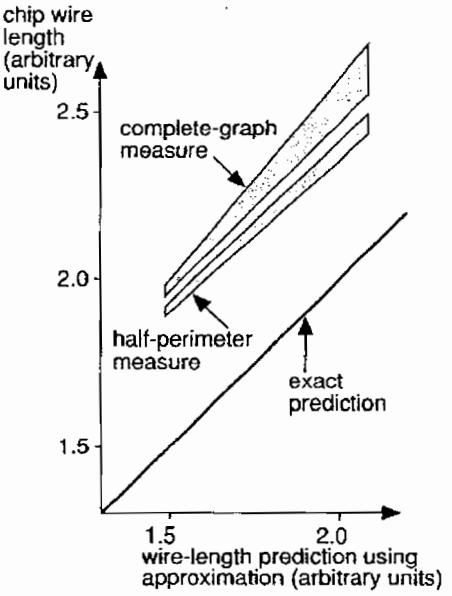


FIGURE 16.22 Correlation between total length of chip interconnect and the half-perimeter and complete-graph measures.

One problem with the measurements we have described is that the MRST may only approximate the interconnect that will be completed by the detailed router. Some programs have a **meander factor** that specifies, on average, the ratio of the interconnect created by the routing tool to the interconnect-length estimate used by the placement tool. Another problem is that we have concentrated on finding estimates to the MRST, but the MRST that minimizes total net length may not minimize net delay (see Section 16.2.8).

There is no point in minimizing the interconnect length if we create a placement that is too congested to route. If we use minimum **interconnect congestion** as an additional placement objective, we need some way of measuring it. What we are trying to measure is interconnect density. Unfortunately we always use the term *density* to mean channel density (which we shall discuss in Section 17.2.2, "Measurement of Channel Density"). In this chapter, while we are discussing placement, we shall try to use the term *congestion*, instead of density, to avoid any confusion.

One measure of interconnect congestion uses the **maximum cut line**. Imagine a horizontal or vertical line drawn anywhere across a chip or block, as shown in Figure 16.23. The number of interconnects that must cross this line is the **cut size** (the number of interconnects we cut). The maximum cut line has the highest cut size.

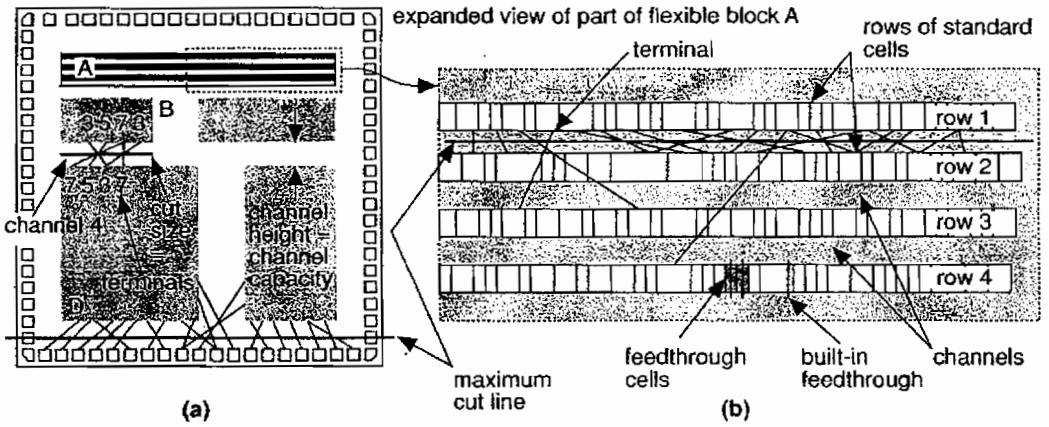


FIGURE 16.23 Interconnect congestion for the cell-based ASIC from Figure 16.11(b). (a) Measurement of congestion. (b) An expanded view of flexible block A shows a maximum cut line.

Many placement tools minimize estimated interconnect length or interconnect congestion as objectives. The problem with this approach is that a logic cell may be placed a long way from another logic cell to which it has just one connection. This logic cell with one connection is less important as far as the total wire length is concerned than other logic cells, to which there are many connections. However, the one long connection may be critical as far as timing delay is concerned. As technology is scaled, interconnection delays become larger relative to circuit delays and this problem gets worse.

In **timing-driven placement** we must estimate delay for every net for every trial placement, possibly for hundreds of thousands of gates. We cannot afford to use anything other than the very simplest estimates of net delay. Unfortunately, the minimum-length Steiner tree does not necessarily correspond to the interconnect path that minimizes delay. To construct a minimum-delay path we may have to route with non-Steiner trees. In the placement phase typically we take a simple interconnect-length approximation to this minimum-delay path (typically the half-perimeter measure). Even when we can estimate the length of the interconnect, we do not yet have information on which layers and how many vias the interconnect will use or how wide it will be. Some tools allow us to include estimates for these parameters. Often we can specify **metal usage**, the percentage of routing on the different layers to expect from the router. This allows the placement tool to estimate RC values and delays—and thus minimize delay.

16.2.4 Placement Algorithms

There are two classes of placement algorithms commonly used in commercial CAI tools: constructive placement and iterative placement improvement. A **constructiv placement method** uses a set of rules to arrive at a constructed placement. The mo commonly used methods are variations on the **min-cut algorithm**. The other com monly used constructive placement algorithm is the **eigenvalue method**. As in sys tem partitioning, placement usually starts with a constructed solution and the improves it using an iterative algorithm. In most tools we can specify the location and relative placements of certain critical logic cells as **seed placements**.

The **min-cut placement** method uses successive application of partitioning [Breuer, 1977]. The following steps are shown in Figure 16.24:

1. Cut the placement area into two pieces.
2. Swap the logic cells to minimize the cut cost.
3. Repeat the process from step 1, cutting smaller pieces until all the logic cells are placed.

Usually we divide the placement area into **bins**. The size of a bin can vary, from a bin size equal to the base cell (for a gate array) to a bin size that would hold several logic cells. We can start with a large bin size, to get a rough placement, and then reduce the bin size to get a final placement.

The **eigenvalue placement algorithm** uses the cost matrix or weighted **connectivity matrix** (eigenvalue methods are also known as **spectral methods**) [Hall, 1970]. The measure we use is a cost function f that we shall minimize, given by

$$f = \frac{1}{2} \sum_{i,j=1}^{n} c_{ij} d_{ij}^2 , \tag{16.6}$$

where $\mathbf{C} = [c_{ij}]$ is the (possibly weighted) connectivity matrix, and d_{ij} is the Euclidean distance between the centers of logic cell i and logic cell j. Since we are going to minimize a cost function that is the square of the distance between logic cells, these methods are also known as **quadratic placement** methods. This type of cost function leads to a simple mathematical solution. We can rewrite the cost function f in matrix form:

$$f = \frac{1}{2} \sum_{i,j=1}^{n} c_{ij} (x_i - x_j)^2 + (y_i - y_j)^2 = \mathbf{x}^T \mathbf{B} \mathbf{x} + \mathbf{y}^T \mathbf{B} \mathbf{y} , \tag{16.7}$$

In Eq. 16.7, **B** is a symmetric matrix, the **disconnection matrix** (also called the Laplacian).

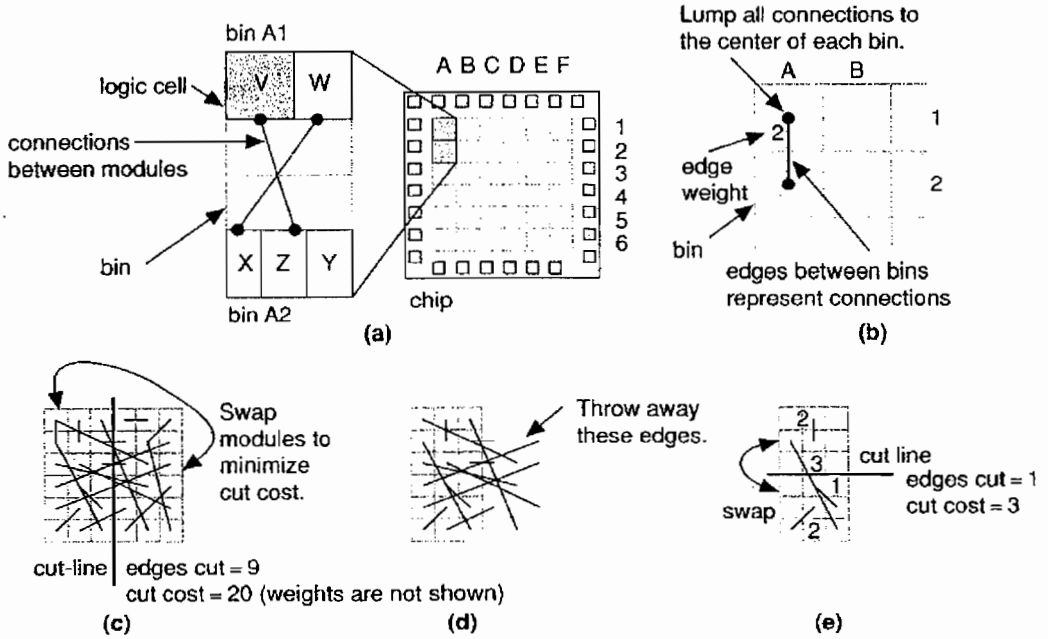


FIGURE 16.24 Min-cut placement. (a) Divide the chip into bins using a grid. (b) Merge all connections to the center of each bin. (c) Make a cut and swap logic cells between bins to minimize the cost of the cut. (d) Take the cut pieces and throw out all the edges that are not inside the piece. (e) Repeat the process with a new cut and continue until we reach the individual bins.

We may express the Laplacian **B** in terms of the connectivity matrix **C**; and **D**, a diagonal matrix (known as the degree matrix), defined as follows:

$$\mathbf{B} = \mathbf{D} - \mathbf{C};\ d_{ii} = \sum_{j=1}^{n} c_{ij},\ i = 1, \ldots, n;\ d_{ij} = 0,\ \ i \neq j. \tag{16.8}$$

We can simplify the problem by noticing that it is symmetric in the x- and y-coordinates. Let us solve the simpler problem of minimizing the cost function for the placement of logic cells along just the x-axis first. We can then apply this solution to the more general two-dimensional placement problem. Before we solve this simpler problem, we introduce a constraint that the coordinates of the logic cells must correspond to valid positions (the cells do not overlap and they are placed on-

grid). We make another simplifying assumption that all logic cells are the same size and we must place them in fixed positions. We can define a vector **p** consisting of the valid positions:

$$\mathbf{p} = [p_1, \ldots, p_n] . \tag{16.9}$$

For a valid placement the x-coordinates of the logic cells,

$$\mathbf{x} = [x_1, \ldots, x_n] , \tag{16.10}$$

must be a permutation of the fixed positions, **p**. We can show that requiring the logic cells to be in fixed positions in this way leads to a series of n equations restricting the values of the logic cell coordinates [Cheng and Kuh, 1984]. If we impose all of these constraint equations the problem becomes very complex. Instead we choose just one of the equations:

$$\sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} p_i^2 . \tag{16.11}$$

Simplifying the problem in this way will lead to an approximate solution to the placement problem. We can write this single constraint on the x-coordinates in matrix form:

$$\mathbf{x}^T\mathbf{x} = \mathrm{P}; \qquad \mathrm{P} = \sum_{i=1}^{n} p_i^2 , \tag{16.12}$$

where P is a constant. We can now summarize the formulation of the problem, with the simplifications that we have made, for a one-dimensional solution. We must minimize a cost function, g (analogous to the cost function f that we defined for the two-dimensional problem in Eq. 16.7), where

$$g = \mathbf{x}^T\mathbf{B}\mathbf{x} \tag{16.13}$$

subject to the constraint:

$$\mathbf{x}^T\mathbf{x} = \mathrm{P} . \tag{16.14}$$

This is a standard problem that we can solve using a Lagrangian multiplier:

$$\Lambda = \mathbf{x}^T\mathbf{B}\mathbf{x} - \lambda[\mathbf{x}^T\mathbf{x} - \mathrm{P}] . \tag{16.15}$$

To find the value of **x** that minimizes g we differentiate Λ partially with respect to **x** and set the result equal to zero. We get the following equation:

$$[\mathbf{B} - \lambda\mathbf{I}]\,\mathbf{x} = \mathbf{0} . \tag{16.16}$$

This last equation is called the **characteristic equation** for the disconnection matrix **B** and occurs frequently in matrix algebra (this λ has nothing to do with scaling). The solutions to this equation are the **eigenvectors** and **eigenvalues** of **B**. Multiplying Eq. 16.16 by $\mathbf{x}^T$ we get:

$$\lambda \mathbf{x}^T \mathbf{x} = \mathbf{x}^T \mathbf{B} \mathbf{x}. \tag{16.17}$$

However, since we imposed the constraint $\mathbf{x}^T\mathbf{x} = P$ and $\mathbf{x}^T\mathbf{B}\mathbf{x} = g$, then

$$\lambda = \frac{g}{P}. \tag{16.18}$$

The eigenvectors of the disconnection matrix **B** are the solutions to our placement problem. It turns out that (because something called the rank of matrix **B** is $n-1$) there is a degenerate solution with all x-coordinates equal ($\lambda = 0$)—this makes some sense because putting all the logic cells on top of one another certainly minimizes the interconnect. The smallest, nonzero, eigenvalue and the corresponding eigenvector provides the solution that we want. In the two-dimensional placement problem, the x- and y-coordinates are given by the eigenvectors corresponding to the two smallest, nonzero, eigenvalues. (In the next section a simple example illustrates this mathematical derivation.)

16.2.5 Eigenvalue Placement Example

Consider the following connectivity matrix **C** and its disconnection matrix **B**, calculated from Eq. 16.8 [Hall, 1970]:

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 2 & -1 & -1 \\ 0 & -1 & 1 & 0 \\ -1 & -1 & 0 & 2 \end{bmatrix}. \tag{16.19}$$

Figure 16.25(a) shows the corresponding network with four logic cells (1–4) and three nets (A-C). Here is a MatLab script to find the eigenvalues and eigenvectors of **B**:

```
C=[0 0 0 1; 0 0 1 1; 0 1 0 0; 1 1 0 0]
D=[1 0 0 0; 0 2 0 0; 0 0 1 0; 0 0 0 2]
B=D-C
[X,D] = eig(B)
```

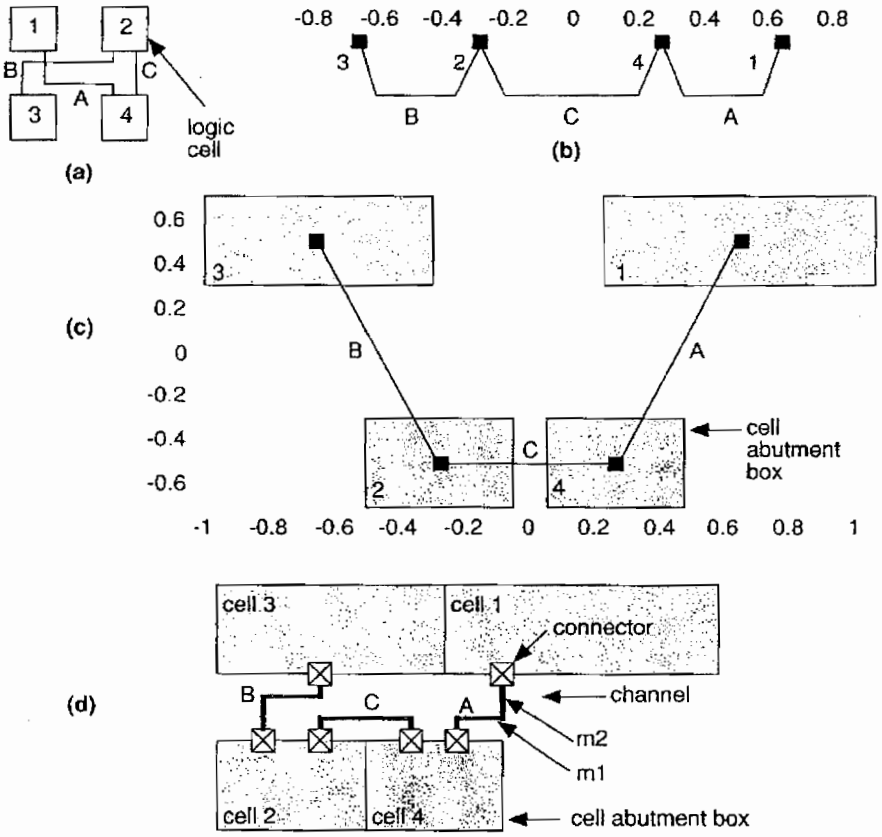


FIGURE 16.25 Eigenvalue placement. (a) An example network. (b) The one-dimensional placement.The small black squares represent the centers of the logic cells. (c) The two-dimensional placement. The eigenvalue method takes no account of the logic cell sizes or actual location of logic cell connectors. (d) A complete layout. We snap the logic cells to valid locations, leaving room for the routing in the channel.

Running this script, we find the eigenvalues of **B** are 0.5858, 0.0, 2.0, and 3.4142. The corresponding eigenvectors of **B** are

$$\begin{bmatrix} 0.6533 & 0.5000 & 0.5000 & -0.2706 \\ -0.2706 & 0.5000 & -0.5000 & -0.6533 \\ -0.6533 & 0.5000 & 0.5000 & 0.2706 \\ 0.2706 & 0.5000 & -0.5000 & 0.6533 \end{bmatrix}. \tag{16.20}$$

For a one-dimensional placement (Figure 16.25b), we use the eigenvector (0.6533, –0.2706, –0.6533, –0.2706) corresponding to the smallest nonzero eigenvalue (which is 0.5858) to place the logic cells along the x-axis. The two-dimensional placement (Figure 16.25c) uses these same values for the x-coordinates and the eigenvector (0.5, –0.5, 0.5, –0.5) that corresponds to the next largest eigenvalue (which is 2.0) for the y-coordinates. Notice that the placement shown in Figure 16.25(c), which shows logic-cell outlines (the logic-cell abutment boxes), takes no account of the cell sizes, and cells may even overlap at this stage. This is because, in Eq. 16.11, we discarded all but one of the constraints necessary to ensure valid solutions. Often we use the approximate eigenvalue solution as an initial placement for one of the iterative improvement algorithms that we shall discuss in Section 16.2.6.

16.2.6 Iterative Placement Improvement

An **iterative placement improvement** algorithm takes an existing placement and tries to improve it by moving the logic cells. There are two parts to the algorithm:

- The selection criteria that decides which logic cells to try moving.
- The measurement criteria that decides whether to move the selected cells.

There are several **interchange** or **iterative exchange** methods that differ in their selection and measurement criteria:

- pairwise interchange,
- force-directed interchange,
- force-directed relaxation, and
- force-directed pairwise relaxation.

All of these methods usually consider only pairs of logic cells to be exchanged. A source logic cell is picked for trial exchange with a destination logic cell. We have already discussed the use of interchange methods applied to the system partitioning step. The most widely used methods use group migration, especially the Kernighan–Lin algorithm. The **pairwise-interchange algorithm** is similar to the interchange algorithm used for iterative improvement in the system partitioning step:

1. Select the source logic cell at random.
2. Try all the other logic cells in turn as the destination logic cell.
3. Use any of the measurement methods we have discussed to decide on whether to accept the interchange.
4. The process repeats from step 1, selecting each logic cell in turn as a source logic cell.

Figure 16.26(a) and (b) show how we can extend pairwise interchange to swap more than two logic cells at a time. If we swap λ logic cells at a time and find a locally optimum solution, we say that solution is **λ-optimum**. The **neighborhood exchange algorithm** is a modification to pairwise interchange that considers only

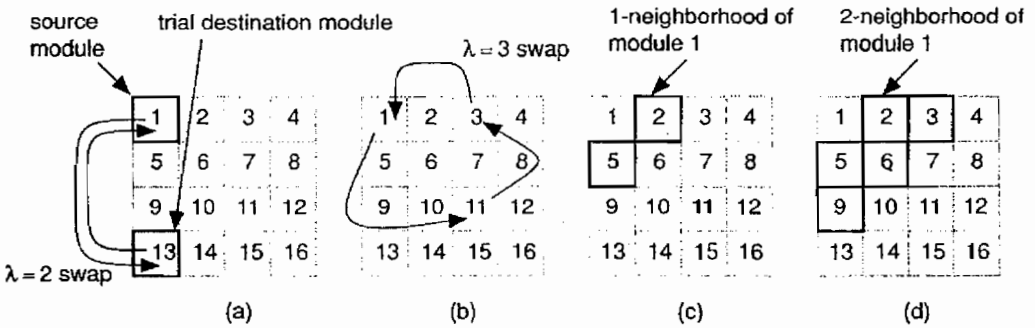


FIGURE 16.26 Interchange. (a) Swapping the source logic cell with a destination logic cell in pairwise interchange. (b) Sometimes we have to swap more than two logic cells at a time to reach an optimum placement, but this is expensive in computation time. Limiting the search to neighborhoods reduces the search time. Logic cells within a distance ε of a logic cell form an ε-neighborhood. (c) A one-neighborhood. (d) A two-neighborhood.

destination logic cells in a **neighborhood**—cells within a certain distance, ε, of the source logic cell. Limiting the search area for the destination logic cell to the **ε-neighborhood** reduces the search time. Figure 16.26(c) and (d) show the one- and two-neighborhoods (based on Manhattan distance) for a logic cell.

Neighborhoods are also used in some of the **force-directed placement methods**. Imagine identical springs connecting all the logic cells we wish to place. The number of springs is equal to the number of connections between logic cells. The effect of the springs is to pull connected logic cells together. The more highly connected the logic cells, the stronger the pull of the springs. The force on a logic cell i due to logic cell j is given by **Hooke's law**, which says the force of a spring is proportional to its extension:

$$F_{ij} = -c_{ij}x_{ij} \tag{16.21}$$

The vector component x_{ij} is directed from the center of logic cell i to the center of logic cell j. The vector magnitude is calculated as either the Euclidean or Manhattan distance between the logic cell centers. The c_{ij} form the connectivity or cost matrix (the matrix element c_{ij} is the number of connections between logic cell i and logic cell j). If we want, we can also weight the c_{ij} to denote critical connections. Figure 16.27 illustrates the force-directed placement algorithm.

In the definition of connectivity (Section 15.7.1, "Measuring Connectivity") it was pointed out that the network graph does not accurately model connections for nets with more than two terminals. Nets such as clock nets, power nets, and global reset lines have a huge number of terminals. The force-directed placement algorithms usually make special allowances for these situations to prevent the largest

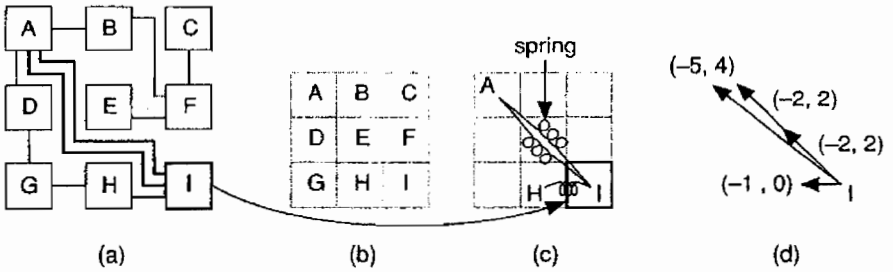


FIGURE 16.27 Force-directed placement. (a) A network with nine logic cells. (b) We make a grid (one logic cell per bin). (c) Forces are calculated as if springs were attached to the centers of each logic cell for each connection. The two nets connecting logic cells A and I correspond to two springs. (d) The forces are proportional to the spring extensions.

nets from snapping all the logic cells together. In fact, without external forces to counteract the pull of the springs between logic cells, the network will collapse to a single point as it settles. An important part of force-directed placement is fixing some of the logic cells in position. Normally ASIC designers use the I/O pads or other external connections to act as anchor points or fixed seeds.

Figure 16.28 illustrates the different kinds of force-directed placement algorithms. The **force-directed interchange** algorithm uses the force vector to select a

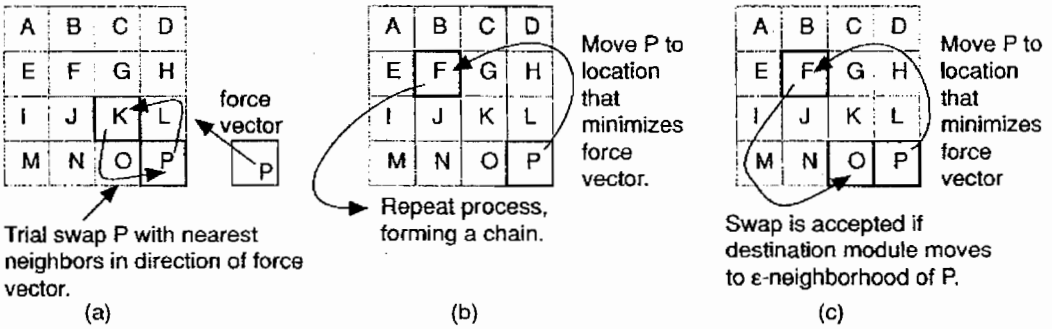


FIGURE 16.28 Force-directed iterative placement improvement. (a) Force-directed interchange. (b) Force-directed relaxation. (c) Force-directed pairwise relaxation.

pair of logic cells to swap. In **force-directed relaxation** a chain of logic cells is moved. The **force-directed pairwise relaxation** algorithm swaps one pair of logic cells at a time.

We reach a force-directed solution when we minimize the energy of the system, corresponding to minimizing the sum of the squares of the distances separating logic cells. Force-directed placement algorithms thus also use a quadratic cost function.

16.2.7 Placement Using Simulated Annealing

The principles of simulated annealing were explained in Section 15.7.8, "Simulated Annealing." Because simulated annealing requires so many iterations, it is critical that the placement objectives be easy and fast to calculate. The optimum connection pattern, the MRST, is difficult to calculate. Using the half-perimeter measure (Section 16.2.3) corresponds to minimizing the total interconnect length. Applying simulated annealing to placement, the algorithm is as follows:

1. Select logic cells for a trial interchange, usually at random.
2. Evaluate the objective function E for the new placement.
3. If ΔE is negative or zero, then exchange the logic cells. If ΔE is positive, then exchange the logic cells with a probability of $\exp(-\Delta E/T)$.
4. Go back to step 1 for a fixed number of times, and then lower the temperature T according to a cooling schedule: $T_{n+1} = 0.9\,T_n$, for example.

Kirkpatrick, Gerlatt, and Vecchi first described the use of simulated annealing applied to VLSI problems [1983]. Experience since that time has shown that simulated annealing normally requires the use of a slow cooling schedule and this means long CPU run times [Sechen, 1988; Wong, Leong, and Liu, 1988]. As a general rule, experiments show that simple min-cut based constructive placement is faster than simulated annealing but that simulated annealing is capable of giving better results at the expense of long computer run times. The iterative improvement methods that we described earlier are capable of giving results as good as simulated annealing, but they use more complex algorithms.

While I am making wild generalizations, I will digress to discuss **benchmarks** of placement algorithms (or any CAD algorithm that is random). It is important to remember that the results of random methods are themselves random. Suppose the results from two random algorithms, A and B, can each vary by ±10 percent for any chip placement, but both algorithms have the same average performance. If we compare single chip placements by both algorithms, they could falsely show algorithm A to be better than B by up to 20 percent or vice versa. Put another way, if we run enough test cases we will eventually find some for which A is better than B by 20 percent—a trick that Ph.D. students and marketing managers both know well. Even single run evaluations over multiple chips is hardly a fair comparison. The only way to obtain meaningful results is to compare a statistically meaningful number of runs for a statistically meaningful number of chips for each algorithm. This same caution applies to any VLSI algorithm that is random. There was a Design Automation Conference panel session whose theme was "Enough of algorithms claiming improvements of 5 %."

16.2.8 Timing-Driven Placement Methods

Minimizing delay is becoming more and more important as a placement objective. There are two main approaches: net based and path based. We know that we can use net weights in our algorithms. The problem is to calculate the weights. One method finds the n most critical paths (using a timing-analysis engine, possibly in the synthesis tool). The net weights might then be the number of times each net appears in this list. The problem with this approach is that as soon as we fix (for example) the first 100 critical nets, suddenly another 200 become critical. This is rather like trying to put worms in a can—as soon as we open the lid to put one in, two more pop out.

Another method to find the net weights uses the **zero-slack algorithm** [Hauge et al., 1987]. Figure 16.29 shows how this works (all times are in nanoseconds). Figure 16.29(a) shows a circuit with **primary inputs** at which we know the **arrival times** (this is the original definition, some people use the term **actual times**) of each signal. We also know the **required times** for the **primary outputs**—the points in time at which we want the signals to be valid. We can work forward from the primary inputs and backward from the primary outputs to determine arrival and required times at each input pin for each net. The difference between the required and arrival times at each input pin is the **slack time** (the time we have to spare). The zero-slack algorithm adds delay to each net until the slacks are zero, as shown in Figure 16.29(b). The net delays can then be converted to weights or constraints in the placement. Notice that we have assumed that all the gates on a net switch at the same time so that the net delay can be placed at the output of the gate driving the net—a rather poor timing model but the best we can use without any routing information.

An important point to remember is that adjusting the net weight, even for every net on a chip, does not theoretically make the placement algorithms any more complex—we have to deal with the numbers anyway. It does not matter whether the net weight is 1 or 6.6, for example. The practical problem, however, is getting the weight information for each net (usually in the form of timing constraints) from a synthesis tool or timing verifier. These files can easily be hundreds of megabytes in size (see Section 16.4).

With the zero-slack algorithm we simplify but overconstrain the problem. For example, we might be able to do a better job by making some nets a little longer than the slack indicates if we can tighten up other nets. What we would really like to do is deal with *paths* such as the critical path shown in Figure 16.29(a) and not just *nets*. Path-based algorithms have been proposed to do this, but they are complex and not all commercial tools have this capability (see, for example, [Youssef, Lin, and Shragowitz, 1992]).

There is still the question of how to predict path delays between gates with only placement information. Usually we still do not compute a routing tree but use simple approximations to the total net length (such as the half-perimeter measure) and then use this to estimate a net delay (the same to each pin on a net). It is not until the routing step that we can make accurate estimates of the actual interconnect delays.

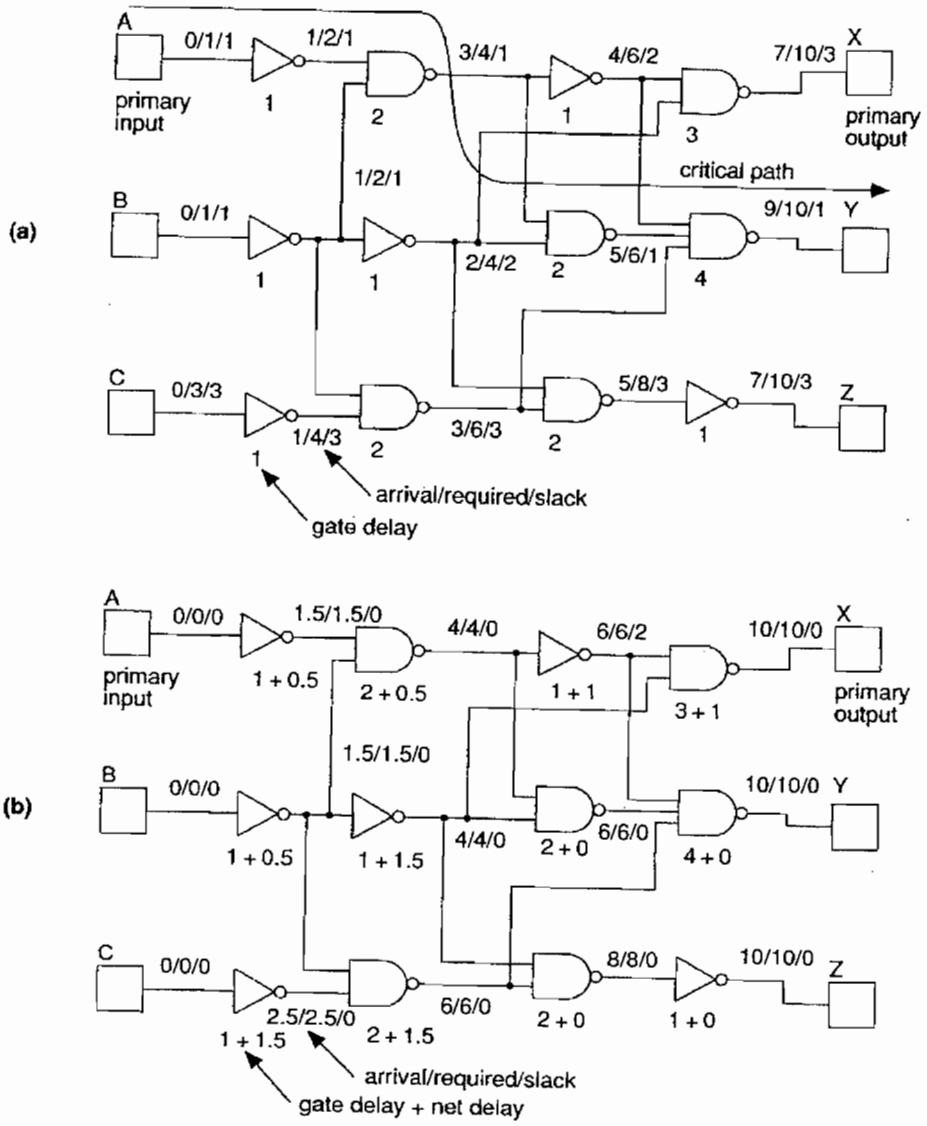
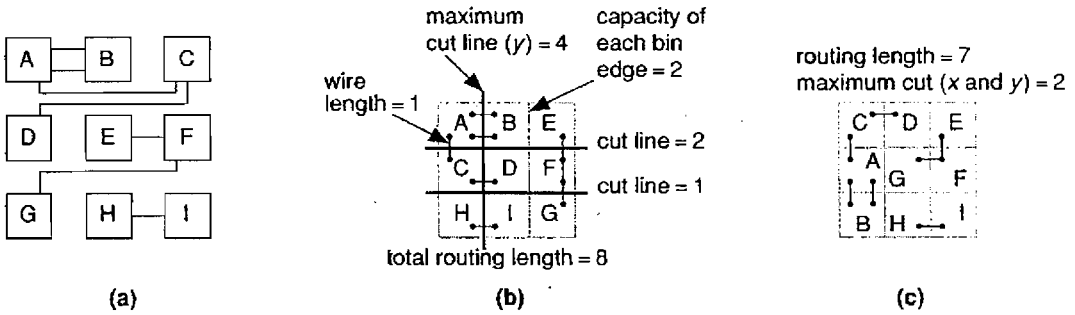


FIGURE 16.29 The zero-slack algorithm. (a) The circuit with no net delays. (b) The zero-slack algorithm adds net delays (at the outputs of each gate, equivalent to increasing the gate delay) to reduce the slack times to zero.

16.2.9 A Simple Placement Example

Figure 16.30 shows an example network and placements to illustrate the measures



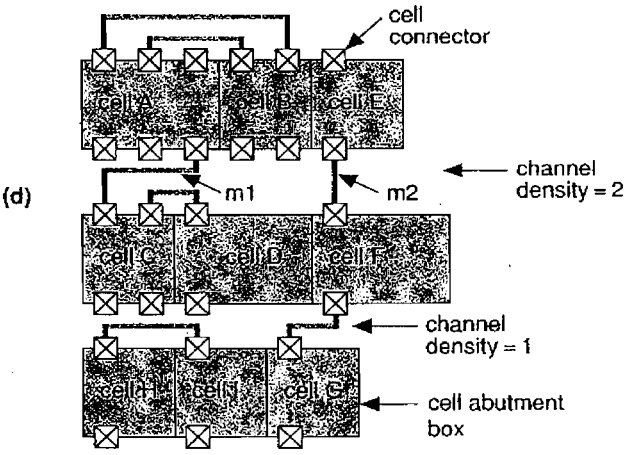


FIGURE 16.30 Placement example. (a) An example network. (b) In this placement, the bin size is equal to the logic cell size and all the logic cells are assumed equal size. (c) An alternative placement with a lower total routing length. (d) A layout that might result from the placement shown in b. The channel densities correspond to the cut-line sizes. Notice that the logic cells are not all the same size (which means there are errors in the interconnect-length estimates we made during placement).

for interconnect length and interconnect congestion. Figure 16.30(b) and (c) illustrate the meaning of total routing length, the maximum cut line in the x-direction, the maximum cut line in the y-direction, and the maximum density. In this example we have assumed that the logic cells are all the same size, connections can be made to terminals on any side, and the routing channels between each adjacent logic cell have a capacity of 2. Figure 16.30(d) shows what the completed layout might look like.

TAB 7

THE AMERICAN HERITAGE DICTIONARY OF THE ENGLISH LANGUAGE

DICT

THE AMERICAN HERITAGE
ONARY
OF THE ENGLISH LANGUAGE

WILLIAM MORRIS, Editor

Published by

HOUGHTON MIFFLIN COMPANY / BOSTON
ATLANTA / DALLAS / GENEVA, ILLINOIS / HOPEWELL, NEW JERSEY / PALO ALTO

Standard Book Numbers: 395–09064–4 (de luxe edition);
395–09066–0 (standard edition; thumb-indexed); 395–09070–9 (school edition);
0–395–20359–7 (new college edition; plain edges);
0–395–20360–0 (new college edition; thumb-indexed)

Library of Congress Catalog Card Number 76–86995

Manufactured in the United States of America

Computer-composed by Inforonics, Inc.
in Maynard, Massachusetts

All-hal-low-mas (ôl′hăl′ō-məs) *n.* Also **All-hal-lows** (ôl′hăl′ōz). **All Saints' Day** *(see).*

all-heal (ôl′hēl′) *n.* Any of several plants reputed to have healing powers, as the **self-heal** *(see).*

al-li-a-ceous (ăl′ē-ā′shəs) *adj.* Characteristic of onions or garlic, especially in odor or taste. [Latin *allium,* garlic (see **allium**) + -ACEOUS.]

al-li-ance (ə-lī′əns) *n.* **1. a.** A formal pact of union or confederation between nations in a common cause. **b.** The nations so conjoined. **2.** Any union, relationship, or connection by kinship, marriage, common interest, or the like. **3.** Any affinity, congruence, or conjunction of quality or kind. **4.** The act of becoming or the state of being allied. [Middle English *alliaunce,* from Old French *aliance,* from *alier,* to ALLY.]

al-lied (ə-līd′, ăl′īd′) *adj.* **1.** Joined; united; confederated. **2.** Of a similar nature; related: *allied studies.* **3.** *Capital* **A.** Of or pertaining to the Allies.

Al-lier (ă′lyā′). A river rising in south-central France and flowing 255 miles north to the Loire.

Al-lies (ăl′īz′, ə-līz′) *pl.n.* **1.** *World War I.* The nations allied against the Central Powers of Europe. They were Russia, France, Great Britain, and later many others, including the United States. **2.** *World War II.* The nations, primarily Great Britain, Russia, and the United States, allied against the Axis.

al-li-ga-tor (ăl′ə-gā′tər) *n.* **1.** Either of two large, amphibious reptiles, *Alligator mississipiensis,* of the southeastern United States, or *A. sinensis,* of China, having sharp teeth and powerful jaws, and differing from crocodiles in having a broader, shorter snout. **2.** Loosely, any crocodilian reptile. **3.** Leather made from the hide of an alligator. **4.** A tool having strong, adjustable jaws, often toothed. [Earlier *alagarto,* from Spanish *el lagarto* : *el,* the, from Latin *ille,* that (see **al-**[1] in Appendix*) + *lagarto,* lizard, from Latin *lacertus,* LIZARD.]

alligator pear. A tree, the **avocado** *(see),* or its fruit. [Folk etymology, variant of AVOCADO (the trees are said to grow in places infested by alligators).]

alligator snapping turtle. A large freshwater turtle, *Macroclemys temminckii,* of the south-central United States, having a rough carapace and a hooked beak. Also called "alligator snapper."

al-lit-er-ate (ə-lĭt′ə-rāt′) *v.* **-ated, -ating, -ates.** —*intr.* **1.** To use alliteration in speech or writing. **2.** To have or contain alliteration. —*tr.* To form or arrange with alliteration. —**al-lit′er-a′tor** (-ə-rā′tər) *n.*

al-lit-er-a-tion (ə-lĭt′ə-rā′shən) *n.* The occurrence in a phrase or line of speech or writing of two or more words having the same initial sound, for example, *wailing in the winter wind.* [From AD- (to) + Latin *littera,* LETTER.]

al-lit-er-a-tive (ə-lĭt′ə-rā′tĭv, -ər-ə-tĭv) *adj.* Of, manifesting, or characterized by alliteration. —**al-lit′er-a′tive-ly** *adv.* —**al-lit′er-a′tive-ness** *n.*

al-li-um (ăl′ē-əm) *n.* Any of various plants of the genus *Allium,* characterized by their pungent odor, and including the onion, leek, chive, garlic, and shallot. [New Latin *Allium,* from Latin *allium, ālium*†, garlic.]

allo-, all-. Indicates divergence, opposition, or difference; for example, **allopatric.** [Greek, other, altered, from *allos,* other. See **al-**[1] in Appendix.*]

al-lo-cate (ăl′ō-kāt′, ăl′ə-) *tr.v.* **-cated, -cating, -cates.** **1.** To designate for a special purpose; set apart. **2.** To distribute according to plan; allot. **3.** To determine the location of; locate. —See Synonyms at **assign.** [Medieval Latin *allocāre,* to place to : Latin *ad-,* toward + *locāre,* to place, from *locus,* place, LOCUS.] —**al′lo-ca-ble** (ăl′ə-kə-bəl) *adj.* —**al′lo-ca′tion** *n.*

al-lo-cu-tion (ăl′ə-kyōō′shən) *n.* A formal and authoritative speech or address, especially one that advises or informs. [Latin *allocūtiō,* from *alloquī* (past participle *allocūtus*), to speak to : *ad-,* to + *loquī,* to speak (see **tolkw-** in Appendix*).]

al-lo-di-um (ə-lō′dē-əm) *n., pl.* **-dia** (-dē-ə). Also **a-lo-di-um.** Land held in absolute ownership, and without obligation or service to any feudal overlord. [Medieval Latin *allodium,* from Frankish *al-ōd-* (unattested). See **ai-**[5] in Appendix.*] —**al-lo′di-al** *adj.* —**al-lo′di-al-ly** *adv.*

al-log-a-my (ə-lŏg′ə-mē) *n. Botany.* **Cross-fertilization** *(see).* [ALLO- + -GAMY.] —**al-log′a-mous** *adj.*

al-lo-graph (ăl′ə-grăf′, -gräf′) *n.* Writing, especially a signature, made by one person for another. [ALLO- + -GRAPH.]

al-lom-er-ism (ə-lŏm′ə-rĭz′əm) *n.* Consistency in crystalline form with variation in chemical composition. [ALLO- + Greek *meros,* part (see **smer-**[2] in Appendix*).] —**al-lom′er-ous** *adj*

al-lom-e-try (ə-lŏm′ə-trē) *n. Biology.* The study of the change in proportion of various parts of an organism as a consequence of growth. [ALLO- + -METRY.] —**al′lo-met′ric** (ăl′ə-mĕt′rĭk) *adj.*

al-lo-morph[1] (ăl′ə-môrf′) *n. Mineralogy.* Obsolete synonym of **paramorph** *(see).* [ALLO- + -MORPH.] —**al′lo-mor′phic** (-môr′fĭk) *adj.* —**al′lo-mor′phism′** *n.*

al-lo-morph[2] (ăl′ə-môrf′) *n. Linguistics.* Any of the variant forms of a morpheme, for example, the phonetic *s* of *cats,* *z* of *dogs,* and *iz* of *horses* are allomorphs of the English morpheme *s.* [ALLO- + -MORPH.] —**al′lo-mor′phic** (-môr′fĭk) *adj.* —**al′lo-mor′phism′** *n.*

al-lo-nym (ăl′ə-nĭm′) *n.* **1.** The name of one person assumed by another, especially by a writer. **2.** A book by such a writer. [French *allonyme* : ALL(O)- + -ONYM.] —**al-lon′y-mous** (ə-lŏn′ə-məs) *adj.* —**al-lon′y-mous-ly** *adv.*

al-lo-path (ăl′ə-păth′). Also **al-lop-a-thist** (ə-lŏp′ə-thĭst) *n.* A person who practices or advocates allopathy.

al-lop-a-thy (ə-lŏp′ə-thē) *n.* Therapy with remedies that produce effects differing from those of the disease treated. Compare **homeopathy.** [German *Allopathie* : ALLO- + -PATHY.] —**al′lo-path′ic** (ăl′ə-păth′ĭk) *adj.* —**al′lo-path′i-cal-ly** *adv.*

al-lo-pat-ric (ăl′ə-păt′rĭk) *adj. Ecology.* Occurring in separate, widely differing geographic areas. Compare **sympatric.** [From ALLO- + Greek *patra,* fatherland, from *patēr,* father (see **peter** in Appendix*).] —**al′lo-pat′ri-cal-ly** *adv.*

al-lo-phane (ăl′ə-fān′) *n.* An amorphous, translucent, variously colored mineral, essentially hydrous aluminum silicate. [Greek *allophanēs,* "appearing otherwise" : ALLO- + -PHANE.]

al-lo-phone (ăl′ə-fōn′) *n. Linguistics.* Any of the variant forms of a phoneme; for example, the aspirated *p* of *pit* and the unaspirated *p* of *spit* are allophones of the English phoneme *p.* [ALLO- + -PHONE.] —**al′lo-phon′ic** (-fŏn′ĭk) *adj.*

all-or-none (ôl′ər-nŭn′) *adj.* Characterized by either complete response or total lack of response or effect, as in neurological action above a threshold.

al-lot (ə-lŏt′) *tr.v.* **-lotted, -lotting, -lots.** **1.** To distribute by lot; apportion. **2.** To give or assign; allocate: *allot three weeks to a project.* —See Synonyms at **assign.** [Middle English *alotten,* from Old French *aloter* : *a-,* from Latin *ad-,* to + *lot,* a portion, lot, from Frankish *lot* (unattested) (see **klēu-** in Appendix*).] —**al-lot′ter** *n.*

al-lot-ment (ə-lŏt′mənt) *n.* **1.** The act of allotting. **2.** That which is allotted. **3.** *Military.* A portion of a serviceman's pay set aside for a member of his family or for insurance.

al-lo-trope (ăl′ə-trōp′) *n.* A structurally differentiated form of an allotropic element. [Back-formation from ALLOTROPY.]

al-lot-ro-py (ə-lŏt′rə-pē) *n.* The existence, especially in the solid state, of two or more crystalline or molecular structural forms of an element. [ALLO- + -TROPY.] —**al′lo-trop′ic** (ăl′ə-trŏp′ĭk), **al′lo-trop′i-cal** *adj.* —**al′lo-trop′i-cal-ly** *adv.*

all′ ot-ta-va (äl ō-tä′və). *Music. Symbol* **8va** **1.** A direction placed above or below notes to be performed an octave higher or lower than written. Also called "all' ottava alta," "all' ottava sopra." **2.** A direction placed below notes to be performed an octave lower than written. In this sense, also called "all' ottava bassa," "all' ottava sotta." [Italian, "at the octave."]

al-lot-tee (ə-lŏt′ē′) *n.* One to whom something is allotted.

all-out (ôl′out′) *adj.* Complete; without reservation; out-and-out: *an all-out effort.*

all-o-ver (ôl′ō′vər) *adj.* Covering an entire surface.

al-low (ə-lou′) *tr.v.* **-lowed, -lowing, -lows.** **1.** To let do or happen; permit. **2.** To acknowledge or admit; concede: *allow the legality of a claim.* **3.** To permit to have. **4.** To make provision for: *allow time for a coffee break.* **5.** To permit the presence of: *No pets allowed.* **6.** To provide (the needed amount): *allow funds in case of emergency.* **7.** To admit; grant: *I allow that to be true.* —**allow for.** To make an allowance or provision for: *allow for bad weather.* —**allow of.** To permit: *a treatise allowing of several interpretations.* [Middle English *allowen,* from Old French *al(l)ouer,* to permit, approve, a blend of: **(a)** Medieval Latin *allocāre,* to assign, ALLOCATE, and **(b)** Latin *allaudāre,* to give praise to : *ad-,* to + *laudāre,* to praise, LAUD.] —**al-low′a-ble** *adj.* —**al-low′a-bly** *adv.*

Synonyms: *allow, permit, let.* These verbs are compared as they mean to grant or to consent to something. *Allow* implies refraining from any hindrance, whereas *permit* suggests authoritative consent. Inherent in both is capacity to prevent an act. *Let* is less strong in implying authority; often it suggests weak consent or failure to prevent something because one is inattentive or not inclined to act.

al-low-ance (ə-lou′əns) *n.* **1.** The act of allowing. **2.** That which is allowed. **3.** A regular provision of money, food, or the like, as to a dependent. **4.** A price reduction granted as in exchange for used merchandise; discount. **5.** A consideration for possibilities or modifying circumstances: *an allowance for breakage.* —*tr.v.* **allowanced, -ancing, -ances.** **1.** To restrict to an allowance. **2.** To put on an allowance.

al-low-ed-ly (ə-lou′ĭd-lē) *adv.* By general admission; admittedly.

al-loy (ăl′oi′, ə-loi′) *n.* **1.** *Metallurgy.* A macroscopically homogeneous mixture or solid solution, usually of two or more metals, the atoms of one replacing or occupying interstitial positions between atoms of the other. **2.** Anything added that lowers value or purity. —*tr.v.* (ə-loi′, ăl′oi′) **alloyed, -loying, -loys.** **1.** *Metallurgy.* To combine (metals) to form an alloy. **2.** To lower purity or value of (a metal) by mixing with a cheaper metal. **3.** To debase by the addition of an inferior element. [Old French *aloi,* from *aloier, aleier,* to alloy, to bind, from Latin *alligāre,* to bind to, ALLY.]

all right. **1.** Satisfactory; average. **2.** Correct. **3.** Uninjured. **4.** Very well; yes. **5.** Without a doubt: *He's a fool, all right!*

all-right (ôl′rīt′) *adj. Slang.* **1.** Dependable; honorable: *an all-right fellow.* **2.** Good; excellent: *an all-right movie.*

all-round (ôl′round′) *adj.* Also **all-a-round** (ôl′ə-round′). **1.** Comprehensive in extent or depth: *all-round vocational training.* **2.** Able to do many or all things well; generally excellent; versatile: *an all-round student.*

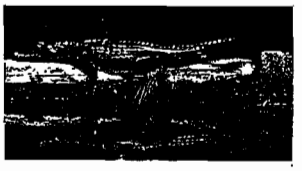
Usage: *All-round* is preferable to the variant form *all-around.* Both are hyphenated and applicable only as adjectives standing before nouns. They should not be confused with *all round* and *all around* used adverbially or prepositionally.

All Saints' Day. November 1, a church festival in honor of all saints. Also called "Allhallowmas," "Allhallows."

all-seed (ôl′sēd′) *n. Botany.* Any of several plants having many seeds, such as **knotgrass** *(see).*

All Souls' Day. November 2, observed by the Roman Catholic Church as a day of prayer for souls in purgatory.

all-spice (ôl′spīs′) *n.* **1.** A tropical American tree, *Pimenta officinalis,* having small white flowers and aromatic berries.



alligator
Alligator mississipiensis

allspice
Branch of allspice with leaves and berries

t tight/th thin, path/*th* this, bathe/ū cut/ûr urge/v valve/w with/y yes/z zebra, size/zh vision/ə about, item, edible, gallop, circus/ à *Fr.* ami/œ *Fr.* feu, *Ger.* schön/ü *Fr.* tu, *Ger.* über/KH *Ger.* ich, *Scot.* loch/N *Fr.* bon. *Follows main vocabulary. †Of obscure origin.

TAB 8

US006519745B1

(12) **United States Patent**
Srinivas et al.

(10) **Patent No.: US 6,519,745 B1**
(45) **Date of Patent: Feb. 11, 2003**

(54) **SYSTEM AND METHOD FOR ESTIMATING CAPACITANCE OF WIRES BASED ON CONGESTION INFORMATION**

(75) Inventors: **Prasanna Venkat Srinivas**, Cupertino, CA (US); **Manjit Borah**, Cupertino, CA (US); **Premal Buch**, Cupertino, CA (US)

(73) Assignee: **Magma Design Automation, Inc.**, Cupertino, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/579,966**

(22) Filed: **May 26, 2000**

(51) **Int. Cl.**[7] **G06F 9/45**; H03K 19/173; H03K 19/177; G01R 31/26

(52) **U.S. Cl.** **716/5**; 326/38; 326/39; 438/17; 716/2; 716/11; 716/13; 716/6

(58) **Field of Search** 702/81; 716/1–21; 257/277; 326/36, 39; 327/99; 438/17

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,185,722	B1 *	2/2001	Darden et al.	716/5
6,311,139	B1 *	10/2001	Kuroda et al.	702/81
6,327,693	B1 *	12/2001	Cheng et al.	716/2

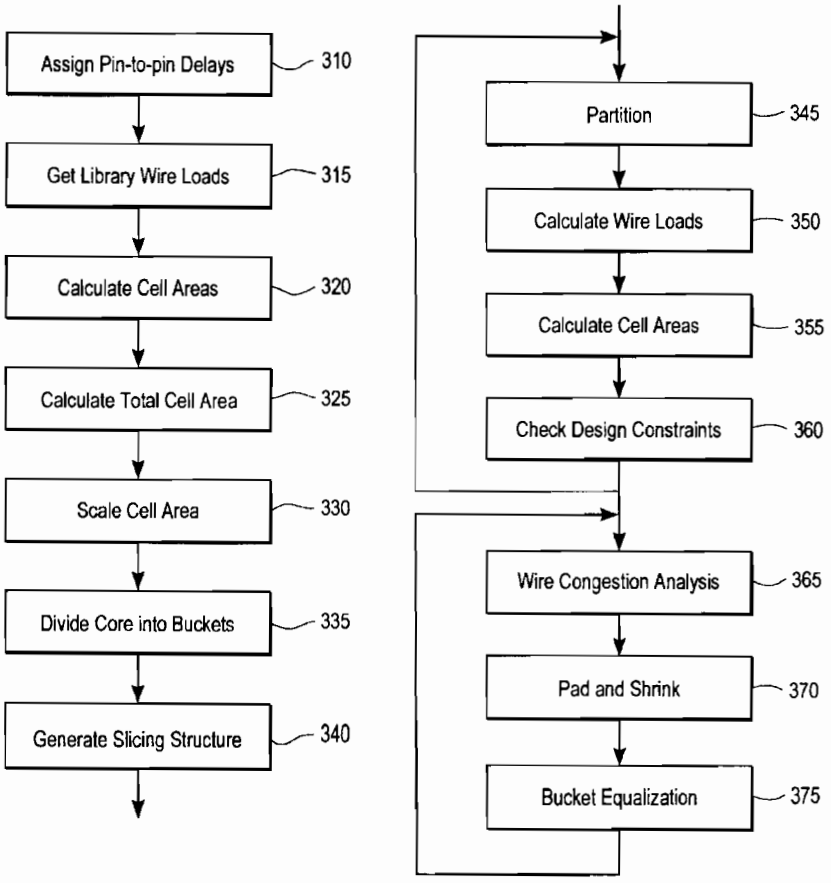
* cited by examiner

Primary Examiner—Vuthe Siek
Assistant Examiner—Helen Rossoshek
(74) *Attorney, Agent, or Firm*—Pillsbury Winthrop LLP

(57) **ABSTRACT**

A system for calculating interconnect wire lateral capacitances in an automated integrated circuit design system subdivides the chip area of a circuit design to be placed and routed into a coarse grid of buckets. An estimate of congestion in each bucket is computed from an estimated amount of routing space available in the bucket and estimated consumption of routing resources by a global router. This congestion score is then used to determine the spacing of the wires in the bucket which is in turn used to estimate the capacitance of the wire segment in the bucket.

8 Claims, 7 Drawing Sheets



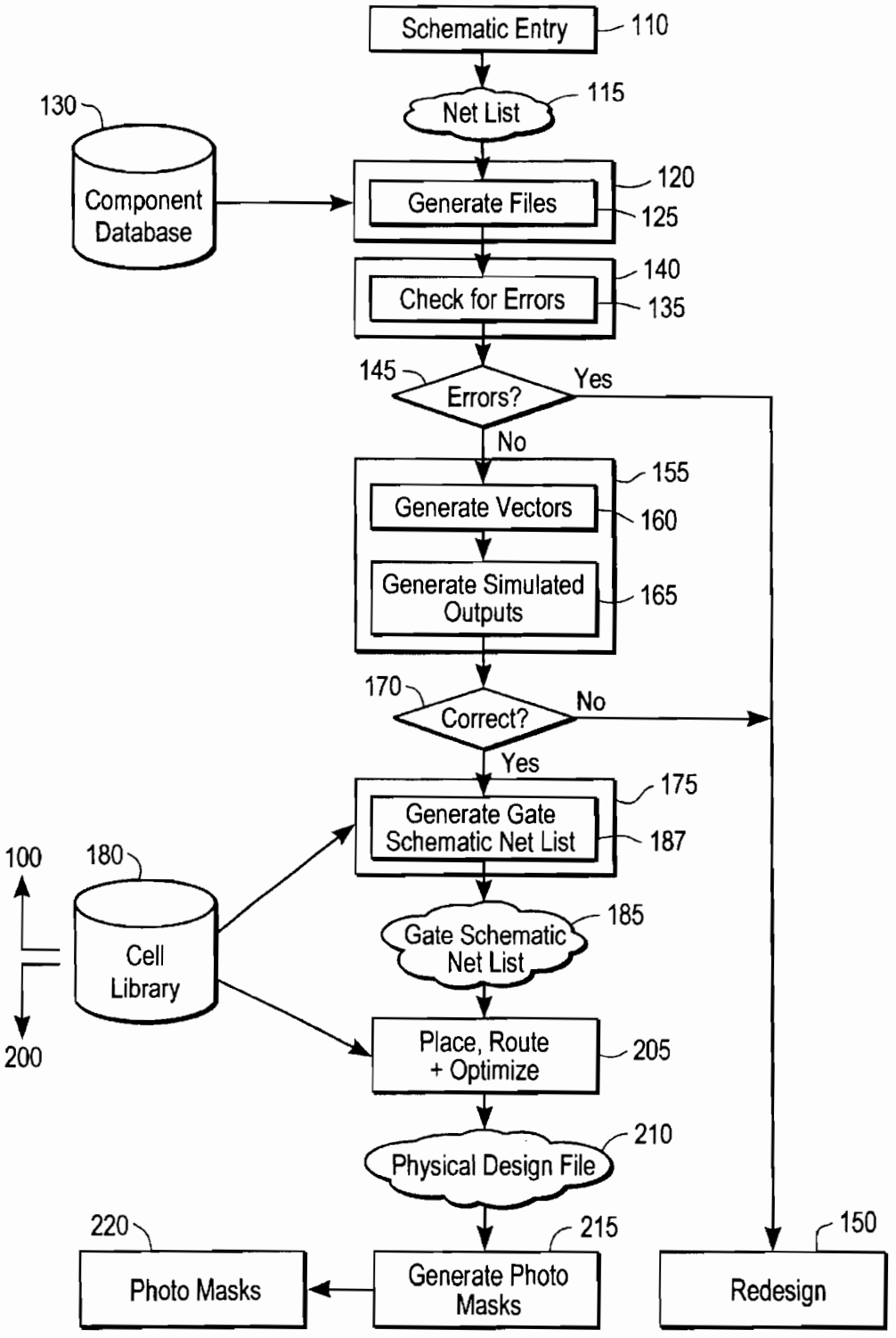

Schematic Entry
110
Net List
115
130
Component Database
Generate Files
120
125
Check for Errors
140
135
145
Errors?
Yes
No
Generate Vectors
155
160
Generate Simulated Outputs
165
170
Correct?
No
Yes
Generate Gate Schematic Net List
175
187
100
180
Cell Library
200
Gate Schematic Net List
185
Place, Route + Optimize
205
Physical Design File
210
220
Photo Masks
Generate Photo Masks
215
Redesign
150

FIG. 1

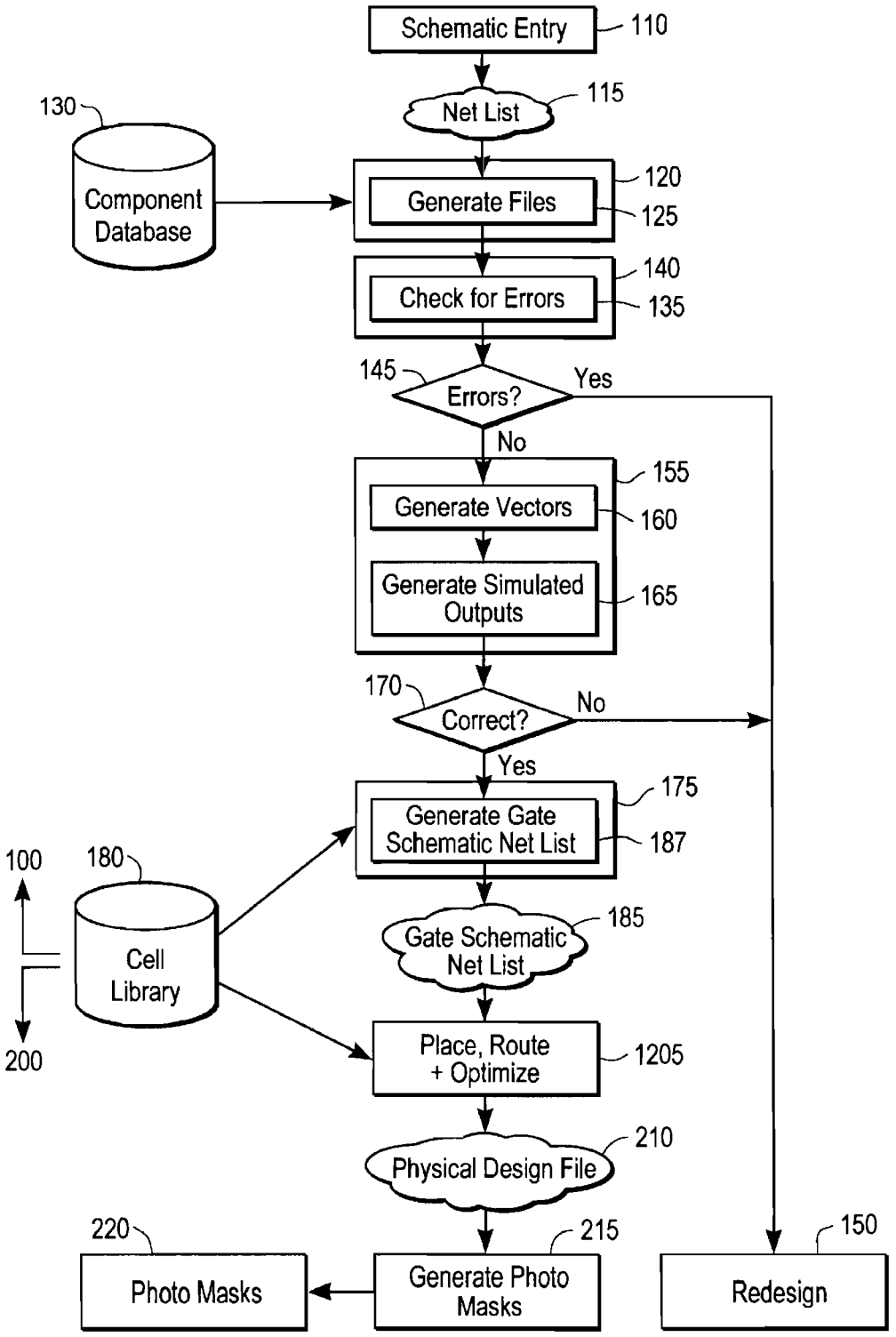

Schematic Entry
110
Net List
115
130
Component Database
Generate Files
120
125
Check for Errors
140
135
145
Errors?
Yes
No
Generate Vectors
155
160
Generate Simulated Outputs
165
170
Correct?
No
Yes
Generate Gate Schematic Net List
175
187
100
180
Cell Library
200
Gate Schematic Net List
185
Place, Route + Optimize
1205
Physical Design File
210
220
Photo Masks
Generate Photo Masks
215
Redesign
150

FIG. 2

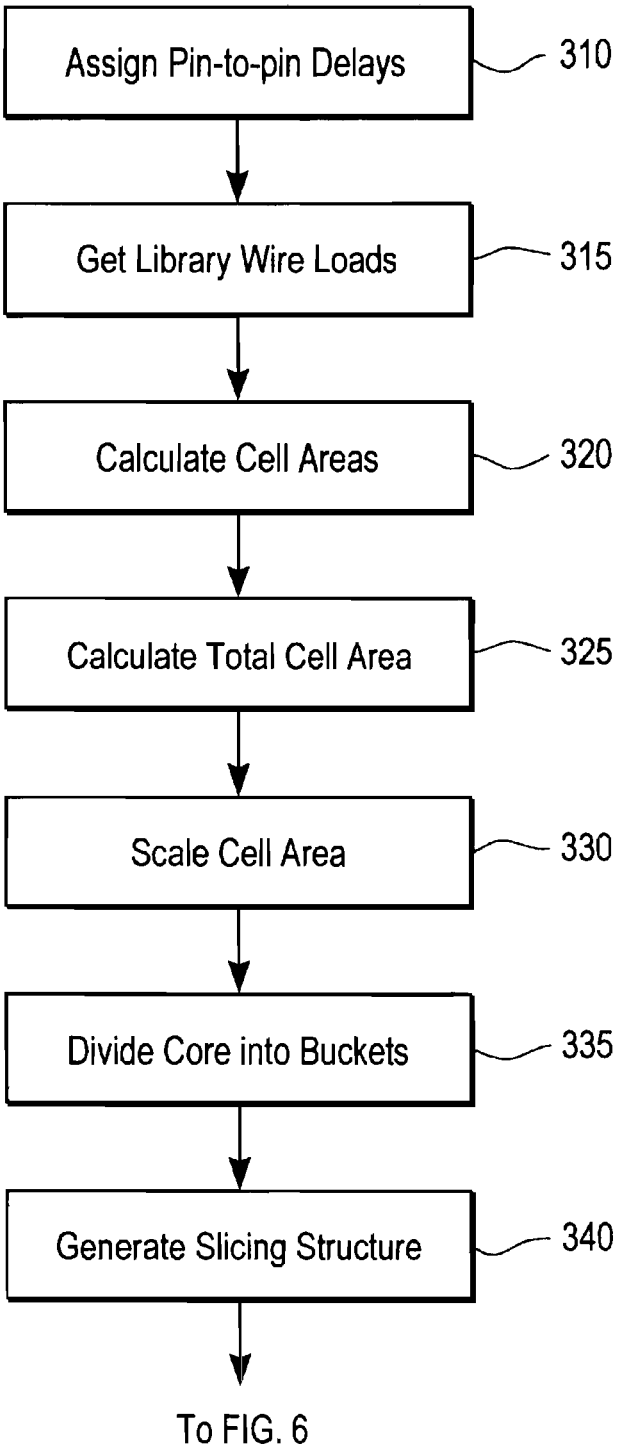

Assign Pin-to-pin Delays
310
Get Library Wire Loads
315
Calculate Cell Areas
320
Calculate Total Cell Area
325
Scale Cell Area
330
Divide Core into Buckets
335
Generate Slicing Structure
340
To FIG. 6

FIG. 3

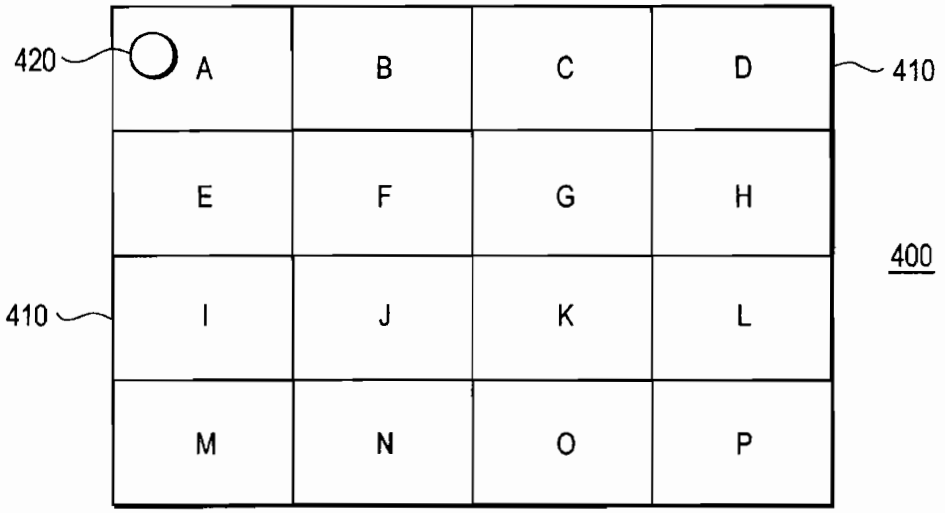

420
A
B
C
D
410
E
F
G
H
400
410
I
J
K
L
M
N
O
P

FIG. 4

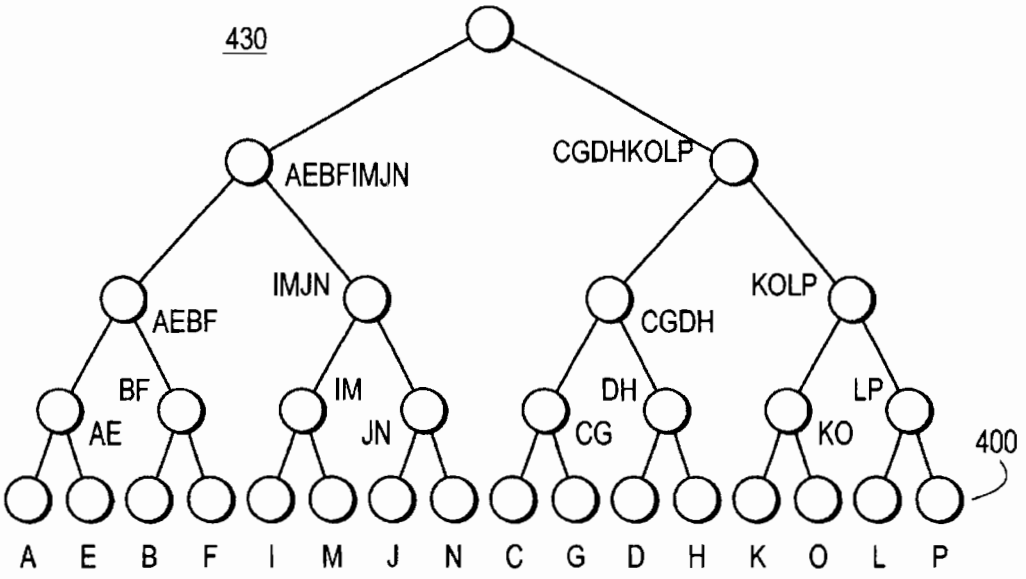

430
AEBFIMJN
CGDHKOLP
AEBF
IMJN
CGDH
KOLP
AE
BF
IM
JN
CG
DH
KO
LP
400
A E B F I M J N C G D H K O L P

FIG. 5

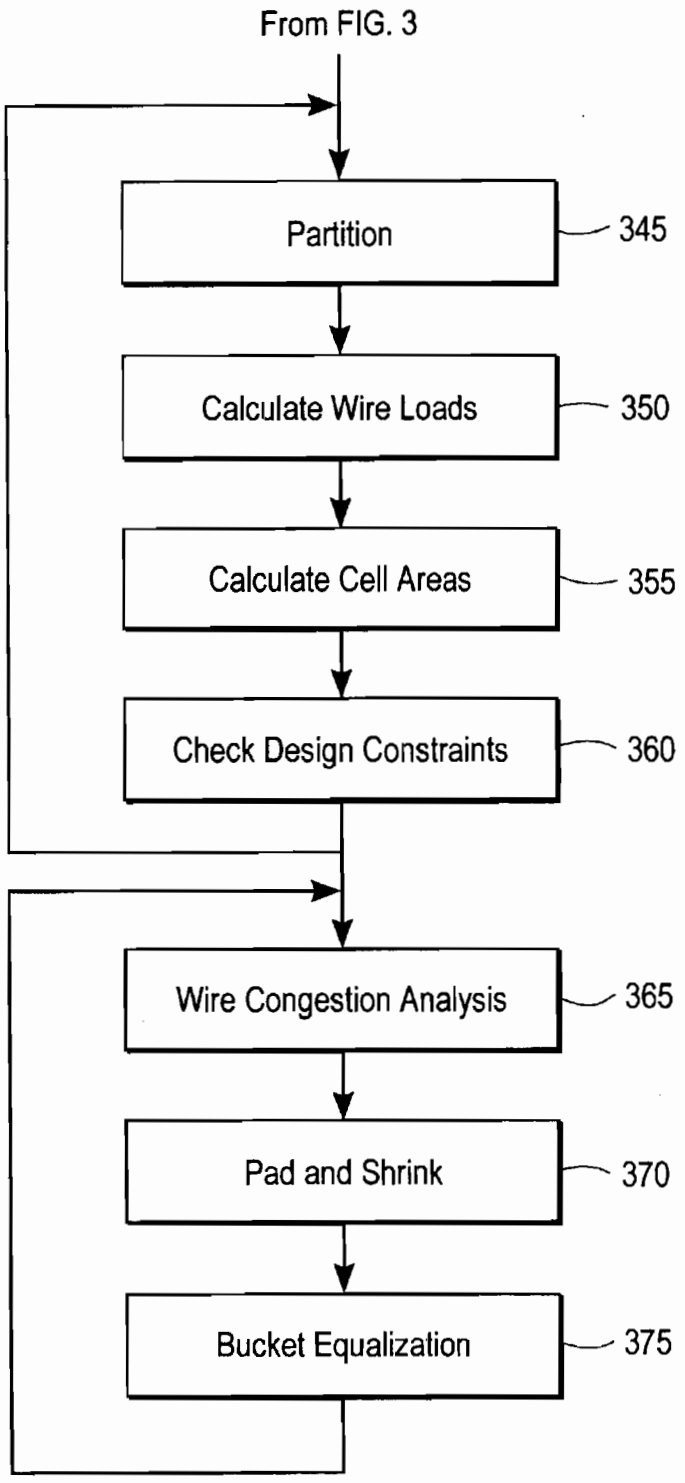

From FIG. 3
Partition
345
Calculate Wire Loads
350
Calculate Cell Areas
355
Check Design Constraints
360
Wire Congestion Analysis
365
Pad and Shrink
370
Bucket Equalization
375

FIG. 6

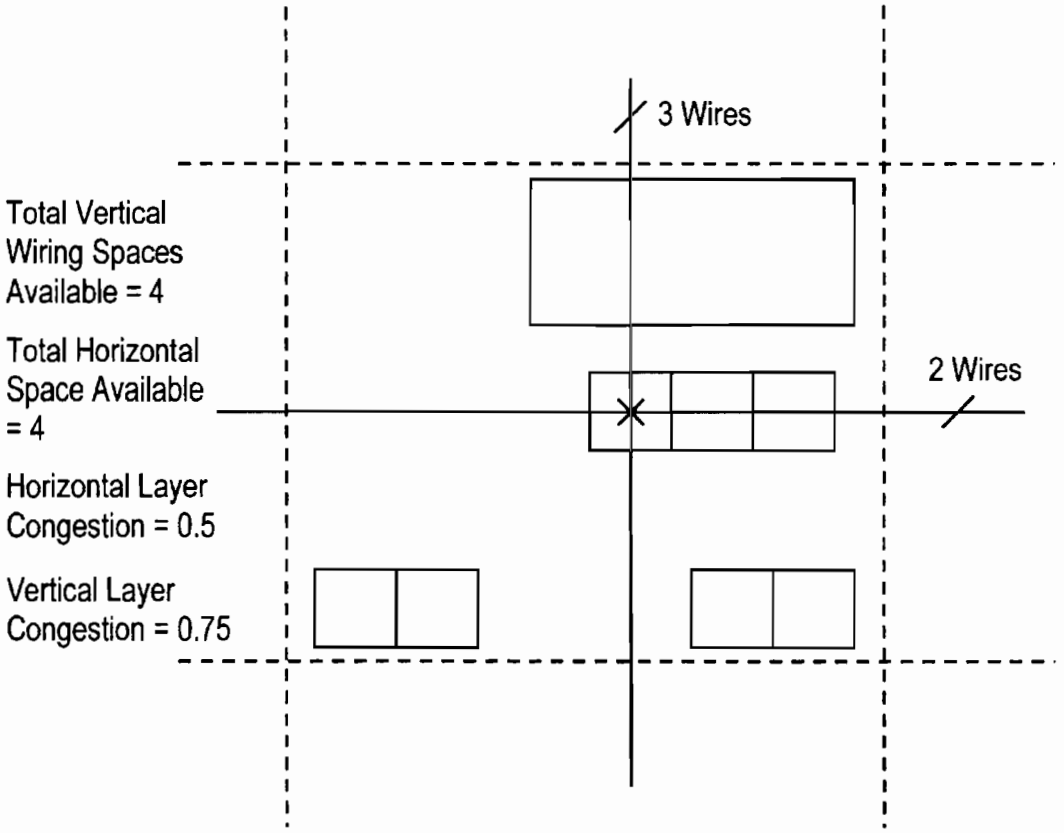

3 Wires
Total Vertical Wiring Spaces Available = 4
Total Horizontal Space Available = 4
2 Wires
Horizontal Layer Congestion = 0.5
Vertical Layer Congestion = 0.75

FIG. 7

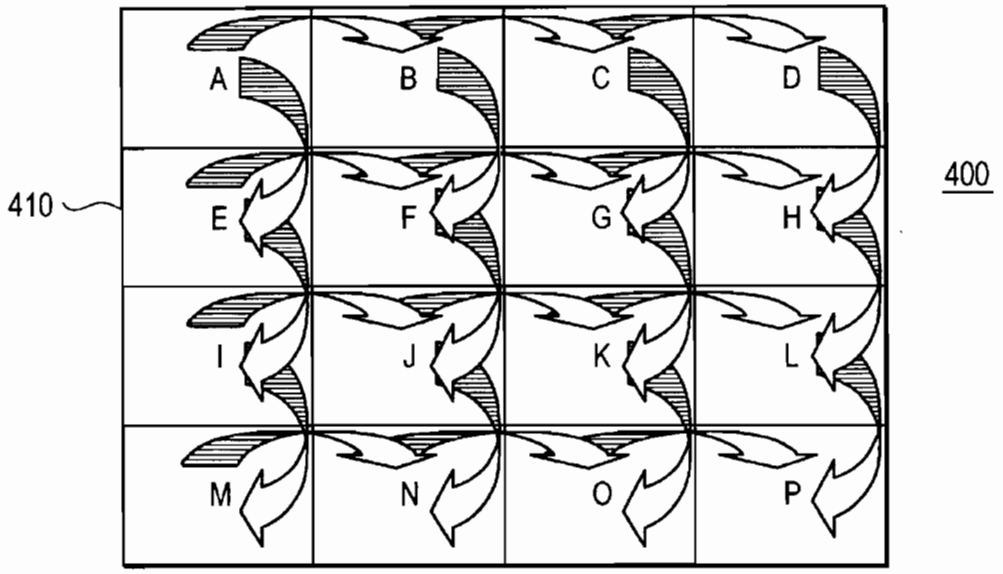

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
410
400

FIG. 8

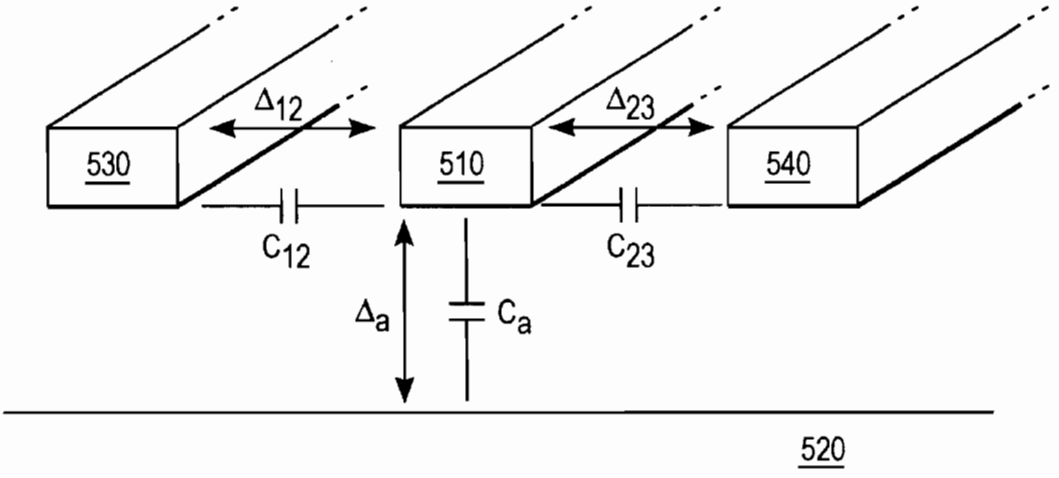

Δ_{12}
Δ_{23}
530
510
540
C_{12}
C_{23}
Δ_a
C_a
520

FIG. 9

SYSTEM AND METHOD FOR ESTIMATING CAPACITANCE OF WIRES BASED ON CONGESTION INFORMATION

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention is directed to digital logic design systems. More particularly, the invention is directed to automated digital logic synthesis and placement systems for integrated circuits, and to performance optimization of digital integrated circuits.

2. Background of the Related Art

Prior art computer aided design (CAD) systems for the design of integrated circuits (ICs) and the like assist in the design thereof by providing a user with a set of software tools running on a digital computer. In the prior art, the process of designing an integrated circuit on a typical CAD system is done in several discrete steps using different software tools.

The design process can be broadly divided into two phases. The initial phase **100** (shown in FIG. **1**) of selecting the right components and connecting them so that the desired functionality is achieved is called logical synthesis. The second phase **200**, in which the selected components are placed within the confines of the chip boundaries and the connecting wires are laid out in order to generate the photographic masks for manufacturing, is called physical synthesis.

First, in the logical synthesis phase **100** a schematic diagram of the integrated circuit is entered interactively in Step **110** to produce a digital representation **115** of the integrated circuit elements and their interconnections. This representation **115** may initially be in a hardware description language such as Verilog or VHDL and then translated into a register transfer level (RTL) description in terms of predesigned functional blocks, such as memories and registers. This may take the form of a data structure called a net list.

Next, a logic compiler **120** receives the net list in Step **125** and, using a component database **130**, puts all of the information necessary for layout, verification and simulation into object files whose formats are optimized specifically for those functions.

Afterwards, in Step **135** a logic verifier **140** preferably checks the schematic for design errors, such as multiple outputs connected together, overloaded signal paths, etc., and generates error indications in Step **145** if any such design problems exist. In many cases, the IC designer improperly connected or improperly placed a physical item within one or more cells. In this case, these errors are flagged to enable her to correct the layout cells in Step **150** so that they perform their proper logical operation.

Also, in Step **135** the verification process preferably checks the cells laid out by hand to determine if multiple design rules have been observed. Design rules may include the timing requirements of the circuit, the area occupied by the final design and parameters derived from other rules dictated by the underlying manufacturing technology. These design rules are provided to integrated circuit designers to ensure that a part can be manufactured with a high degree of yield. Most design rules include hundreds of parameters and, for example, include pitch between metal lines, spacing between diffusion regions in the substrate, sizes of conductive regions to ensure proper contacting without electrical short circuiting, minimum widths of conductive regions, pad sizes, and the like. If a design rules violation is identified in Step **150**, this violation is preferably flagged to the IC designer so that she can properly correct the cells so that they are in accordance with the design rules in Step **150**.

Then, using a simulator **155** the user of the CAD system may prepare a list of vectors representing real input values to be applied to a simulation model of the integrated circuit in Step **160**. This representation may be translated into a form which is better suited to simulation. This representation of the integrated circuit is then operated upon by the simulator which produces numerical outputs analogous to the response of a real circuit with the same inputs applied in Step **165**. By viewing the simulation results, the user may then determine in Step **170** if the represented circuit will perform correctly when it is constructed. If not, she may re-edit the schematic of the integrated circuit, re-compile it and re-simulate it in Step **150**. This process is performed iteratively until the user is satisfied that the design of the integrated circuit is correct.

Then, the human IC designer may present as input to a logic synthesis tool **175** a cell library **180** and a behavioral circuit model. The behavioral circuit model is typically a file in memory which looks very similar to a computer program, and the model contains instructions which logically define the operation of the integrated circuit. The logic synthesis tool **175** maps the instructions from the behavioral circuit model to one or more logic cells from the library **180** to transform the behavioral circuit model to a gate schematic net list **185** of interconnected cells in Step **187**. The gate schematic net list **185** is a database having interconnected logic cells which perform a logical function in accordance with the behavioral circuit model instructions. Once the gate schematic net list **185** is formed, it is provided to a place and route tool **205** to begin the second phase of the design process, physical synthesis.

The place and route tool **205** is preferably then used to access the gate schematic net list **185** and the library cells **180** to position the cells of the gate schematic net list **185** in a two-dimensional format within a surface area of an integrated circuit die perimeter. The output of the place and route step may be a two-dimensional physical design file **210** which indicates the layout interconnection and two-dimensional IC physical arrangements of all gates/cells within the gate schematic net list **185**. From this, in Step **215** the design automation software can create a set of photographic masks **220** to be used in the manufacture of the IC.

One common goal in chip design involves timing performance. The timing performance of the chip is determined by the time required for signals to propagate from one register to another. Clock signals driven at a certain frequency control storage of data in the registers. The time required for a signal to propagate from one register to another depends on the number of levels of cells through which the signal has to propagate, the delay through each of the cells and the delay through the wires connecting these cells. The logic synthesis phase **100** influences the number of levels and the propagation delay through each cell because in it the appropriate components are selected, while the physical synthesis **200** phase affects the propagation delay through the wires.

During the process of timing optimization during physical design in Step **205**, circuit timing is evaluated based on an initial placement and selection of cell strengths. The feedback from the timing analysis is used to drive repeated improvements to the placement software and the selection of the strengths of the cells. The automation software may also perform buffering on some parts of the circuit to optimize

the timing performance by inserting repeater cells, i.e., buffers, to speed up certain paths. Preferably, the optimization software tentatively applies one such modification, evaluates the timing and other constraints (such as design rules dictating capacitance limits) to determine if the step is acceptable and then makes the change permanent if it is deemed acceptable.

The interconnection of the cells in the placing and routing of Step **205** generally involves interconnect wiring having between two and seven metal layers. The delay through an interconnect wire depends on the capacitance of the wire, its resistance and, to a lesser extent, the inductance of the wires. The capacitance of a wire **510** (see FIG. **9**) consists mainly of the capacitance C_a due to the overlap of the wire with the layer **520** above or below, called the area capacitance C_a, and the capacitance due to the overlap along the side walls with other signal wires **530** and **540** adjacent to it, called the lateral capacitance $C_L=C_{12}+C_{13}$. The capacitance of a given wire such as wire **510** can be calculated on a case-by-case basis as is known in the art, and will primarily depend on the wire dimensions D_W and D_T as well as the distance D_L of the wire **510** from the other layers **520** and the distance D_{12} and D_{23} of the wire **510** from the other wires **530** and **540**. In deep sub-micron manufacturing technologies the widths of the wires are becoming thinner and thinner, making them tall and narrow. As a result, under current development trends the lateral capacitance C_L is becoming a dominant component of the total wire capacitance.

During the process of timing optimization during physical design, circuit timing is evaluated based on an initial placement and selection of cell strengths. The feedback from the timing analysis is used to drive the repeated improvements to the placement software and the selection of the strengths of the cells. Since physically laying out all the wires without violating design rules and maintaining good delays is a very time consuming step, a Steiner tree-based topology is used to estimate the area and delay due to the wires based on the current cell placement.

An Elmore delay model is commonly used to compute the wire delay based on the Steiner tree-based topology. However, computing the capacitance of the wires based on the Steiner tree topology is difficult, because nothing is known about the spacing of the wires and the adjacency of different signals at this point in design. Existing approaches use the worst case scenario, and assume that there are wires adjacent to the signal wire in consideration and thus tend to over-estimate the capacitance. Since the optimization software depends on the feedback from the timing analysis, accurate estimation of the capacitance is crucial to the success of the optimizations.

SUMMARY OF THE INVENTION

In view of the above problems of the prior art, it is an object of the present invention to provide a method of estimating the effect of adjacent wires on the capacitance of a signal wire in a circuit design which provides estimates superior to prior art techniques.

It is another object of the present invention to provide a method of estimating the effect of adjacent wires on the capacitance of a signal wire in a circuit design which provides estimates for better than worst case scenarios.

One of the indicators for the expected spacing of the wires in the final routed circuit is the density of the interconnect wires at a given point, i.e., the wires' congestion. The greater the number of wires that passes through a given bucket, the greater the density of wires in that bucket will be and as a result the spacing tends to be smaller to fit all the wires. It would be advantageous to make use of the congestion in a circuit to derive an early estimate of the capacitance.

Thus, it is yet another object of the present invention to provide a method of estimating the effect of adjacent wires on the capacitance of a signal wire in a circuit design which provides estimates based on an estimate of congestion in the design.

The above objects are achieved according to an aspect of the invention by subdividing the chip area of a circuit design to be placed and routed into a coarse grid of buckets. An estimate of congestion in each bucket is computed from an estimated amount of routing space available in the bucket and estimated consumption of routing resources by a global router. This congestion score is then used to determine the spacing of the wires in the bucket which is in turn used to estimate the capacitance of the wire segment in the bucket.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other objects, features and advantages of the present invention are better understood by reading the following detailed description of the preferred embodiment, taken in conjunction with the accompanying drawings, in which:

FIG. **1** is a flowchart of an integrated circuit design process according to the prior art;

FIG. **2** is a flowchart of an integrated circuit design process according to a preferred embodiment of the present invention;

FIGS. **3** and **6** are a flowchart showing a place and route process according to the preferred embodiment;

FIG. **4** shows a coarse grid with buckets in a circuit used with the preferred embodiment;

FIG. **5** shows a slicing structure used in the preferred embodiment;

FIG. **7** shows a cell upon which congestion calculations are performed according to the preferred embodiment;

FIG. **8** shows a bucket mincut process according to the preferred embodiment; and

FIG. **9** shows the major components in routing wire capacitance.

DETAILED DESCRIPTION OF PRESENTLY PREFERRED EXEMPLARY EMBODIMENTS

In one model, the delay through a single logic gate can be represented as

$$d=g\cdot h+p \tag{1}$$

where d is the delay, g is a parameter called the "logical effort" of the gate, h is a parameter called the "electrical effort" of the gate, and p is the parasitic or fixed part of the delay g, in turn, is defined by

$$g_{gate}=\frac{R_{gate\_min}C_{Gate\_min}}{R_{inv\_min}C_{inv\_min}} \tag{2}$$

where gate_min refers to a minimum-sized gate and inv_min to a minimum-sized inverter. h, in turn, is defined by

$$h = \frac{c_{out}}{c_{in}} \quad (3)$$

where c_{out} is the capacitance out of the gate and c_{in} is the capacitance into the gate.

In a constant delay approach to cell placement, the pin-to-pin delay of each cell is fixed early on in the optimization flow. This delay is maintained independently of the load a cell drives. In order to keep delay constant, the size of the cell is adjusted according to the load that it drives. As a result, the area of each cell in the design varies with the load that it drives. The area of each cell is

$$a=b+s\cdot c_{out} \quad (4)$$

where b and s are constants related to the logic of the cell and the chosen constant delay for the cell. Thus, the total area of the netlist is, in matrix notation for plural gates,

$$A=B+SC \quad (5)$$

Approximating the input load at each pin of the cell by c_k/h_k, the total load at the output of a cell i is

$$c_i = \frac{c_j}{g_j} + \frac{c_k}{g_k} \ldots + d_i \quad (6)$$

or, alternatively,

$$c_i = \sum_{\text{fanout}} \frac{c_k}{h_k} + d_i \quad (7)$$

where d_i is the wire load. That is, the total load at the output of cell i is the sum of all its fanout loads plus the load of the wire connecting the cell to its fanouts. In matrix notation for plural gates,

$$C=HC+D \quad (8)$$

$$(I-H)C=D \quad (9)$$

Setting G=I−H,

$$GC=D \quad (10)$$

$$C=G^{-1}D \quad (11)$$

where C is the output capacitance of all gates in the circuit and D is the wire load. Thus, according to the last equation above, the output load of the cells in the circuit can be found once the placement is known. Then, the size of each cell can be found to keep its delay constant. The area of each cell in the netlist denoted by A is

$$A=K_1+K_2^TC \quad (12)$$

and substituting Equation (10) gives

$$A=K_1+K_2^TG^{-1}D \quad (13)$$

Since the load of each wire can be represented as $d=u\cdot l$, where d is the wire load, u is the load per unit length of wire and **1** is the total length of the wire,

$$A=K_1+\mu K_2^TG^{-1}L \quad (14)$$

and combining constant terms,

$$A=K_1+WL \quad (15)$$

Thus, in order to minimize the circuit area one can minimize WL, where the matrix W may be viewed as a set of weights of the wire lengths L.

Generally, the cell is modeled as a rectangle, with the height of the rectangle being the height of a standard cell row. Thus, the width and therefore the area of the cell are functions of the load.

The preferred embodiment of the present invention processes a data structure representative of the circuit being placed and routed. Preferably, this is done on a digital computer as is known in the art. The data structure may be a netlist or other suitable structure known in the art; however, it is preferably a data model of the type disclosed in the U.S. patent application Ser. No. 09/300,540 to Van Ginneken et al; however, other simpler structures may be used as well.

The overall flow of a place and route process **1205** (see FIG. **2**) according to a preferred embodiment of the present invention is shown in FIGS. **3** and **6**. Since a design constraint of the placement process is that the delay across a net be constant, in the preferred embodiment the area of a cell is dependent on the load it drives. In turn, the load of a wire is not known with certainty until the placement process is finished. Thus, to make an initial placement of cells within the core some initial estimations are preferably made. Each cell is assigned a pin-to-pin constant delay in Step **310**. Appropriate techniques will be readily apparent to those skilled in the art; however, pin delay assignment is preferably done according to the technique described in the U.S. patent application Ser. No. 09/300,666. Throughout the placement process, this delay will be maintained constant and the area of the cell varied according to the load it drives in order to achieve the assigned delay.

To make the initial cell placement, the area of each cell is calculated in Step **320** using wire loads obtained from the cell library in Step **315** and substituted into Equation 15. Although cells of varying power levels are available only in discrete steps in the target cell library, this phase of the technique proceeds as if a continuous spectrum of cell power levels are available and selects a cell from the library closest to the size ultimately selected as one of the final steps of the process.

The total cell area A_{total} is determined by adding up the areas of all the cells in Step **325**, and the sizes of the cells are scaled to achieve a target percentage of core utilization in Step **330**. Based on this, standard cell rows are created.

More specifically, the core **400** where the cells are placed is divided into coarse placement regions called buckets **410** as shown in FIG. **4**. Each bucket **410** is a small rectangular region within the core **400**. Buckets **410** have equal dimensions but the placeable area within a bucket **410** depends on the presence of blockages such as macros in the bucket **410**. A bucket **410** can accommodate about fifty average-sized standard cells **420**. Then, in Step **340** a slicing structure or binary tree **430** is built whose leaves **440** are the coarse buckets **410**. For example, a core **400** having a 4×4 matrix of buckets **410** imposed thereon (of course, in practice there will be a much greater number of buckets **410**) as shown in FIG. **4** can be represented by the slicing structure **430** shown in FIG. **5**.

Cells **420** are assigned to the buckets **410** so that the total area of cells **420** within each bucket **410** closely matches the area of that bucket **410**. This is done by an iterative bipartitioning of the data model. First, a horizontal or vertical cut of the core **400** is chosen. The total area available on each side of the partition is computed. Cells **420** are divided using quadratic placement (see below) and a mincut technique

(see, e.g., Fiduccia et al., "A Linear Time Heuristic for Improving Network Partitions", ACM/IEEE Design Automation Conference, 1982, pp. 175–81) on each side so that total wire length is minimized. This iteration continues until a desired resolution, e.g., a bucket **410**, is reached.

Later, each cell **420** is assigned to one of the buckets **410** using a partitioning technique in Step **345** as shown in FIG. **6**. The second placement is done instead of a single cell-level placement because the first placement is done with a coarse grid, i.e., buckets containing hundreds of cells. Here, the cells are placed into their corresponding buckets. In the later stage when other optimizations such as choosing the size and adding repeaters are done and the netlist is more stable, the second, detailed placement is done. This placement step places a cell in its actual location. In this stage, some cells may move from their originally-assigned bucket to a neighboring bucket to make room for the cells in a highly populated bucket in order to reduce congestion.

A good placement of cells **420** is one that can be easily routed and satisfies the given timing constraints for the logic circuit. Quadratic placement, and in particular Gordian quadratic placement, finds a legal placement while minimizing the total squared wire length in the circuit and is the placement technique preferably used. Gordian quadratic placement is well-known in the art as shown by, e.g., Klienhans et al., "GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization", IEEE Trans on Computer-Aided Design, v. 10, n. 3 (Mar. 1991), pp. 356–365, and for simplicity will be generally described below.

The problem is independently solved for the x and y coordinates. Briefly describing the process for the x coordinates (the process for the y coordinates is similar), quadratic placement solves the following equation subject to a constraint Hx=t (to account for physical realities such as overlapping cells and the like) to minimize total wire length during placement:

$$\tfrac{1}{2}(\Sigma c_{ij}(x_i-x_j)^2+\Sigma c_{ij}(x_i-b_j)^2) \quad (16)$$

$$\tfrac{1}{2}x'Ax-x'd+\text{constant} \quad (17)$$

x is the location of cells **420** and star nets. Star nets are nets with more than fifteen pins. A star net is treated like a cell **420**. All cells **420** attached to a star net are considered to be attached to the center of the net through a two-pin net. Star nets are used to reduce the number of fill-ins in the matrix A. The weight of a net k_i is 2/(number of pins). The weight of a net connecting a cell **420** to the center of a star net is 1.b has the locations of fixed points. Fixed points are pins of pads or macros. The diagonal elements of A are non-zero and are computed as follows:

$$a_{ii}=\text{SUM}k_i \quad (18)$$

Any cell **420** connecting to cell ii through a non-star net and a star net connecting to a cell i contribute to the summation. The element a_{ij} is non-zero if cells i and j are connected through a net.

$$a_{ij}=-\text{SUM } k_i \quad (19)$$

The contribution comes from the nets connecting cells i and j.

$$d_i=\text{SUM } b_j k_i \quad (20)$$

The contribution comes from all constant pins attached to cell i. The x coordinates for a placement that minimizes the total wire length is obtained by solving

$$Ax=d \quad (21)$$

The initial constraints for quadratic placement assumes the center of mass for all cells **420** on the chip is the center of the chip. If the area of each cell is a_i, $\Sigma a_i x_i = X_{center}$ forms the first constraints for quadratic placement.

The place and route software then performs global routing on the placed cells **420**. Global routing is a coarse level routing that uses Steiner tree-based topologies to connect the bucket centers. The use of Steiner trees in such contexts is well-known in the art; see, for example, Borah et al., "An Edge-Based Heuristic for Steiner Routing", IEEE Transactions on Circuits and Systems 1563–68 (1993) and Griffith et al., "Closing the Gap: Near-Optimal Steiner Trees in Polynomial Time", IEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, v. 13,n. 11 (Nov. 1994) 1351–65 (both incorporated herein by reference) and will not be described in greater detail herein.

When the global routing software adds a wire through a bucket (either during the initial Steiner tree calculation or during recalculations as described below), the budget for the appropriate layer is adjusted to reflect the use of one routing resource from the bucket. More specifically, in order to concisely describe congestion in a bucket, the available routing space in each routing layer is estimated in advance by the computer and assigned to the bucket as a budget for that layer. Every time a wire is routed through the bucket in a particular routing layer, the budget for that layer is adjusted to reflect that one routing space has been spent. The lower the number of available spaces, the greater the congestion in the bucket in the given routing layer.

FIG. **7** illustrates an example of a bucket with three routing wires running through it in the vertical direction in one layer and two routing wires running through it in the horizontal direction in another layer. The congestion score for a bucket is defined as the ratio of the routing resources used so far to the total routing resources available in the bucket. For example, the vertical layer of the bucket of FIG. **7** has a total of four routing spaces (its actual dimensions will be larger than this due to design rule considerations as described in greater detail below), and three vertical wires are going through the bucket. Thus, this bucket has a congestion score of 0.75 along the layer in the vertical routing direction. Similarly, the horizontal routing direction has 2 wires going through it, resulting in a congestion score of 0.5.

When the capacitance for a wire is estimated, e.g., during optimization operations such as inserting repeaters, choosing appropriate strengths, etc., the congestion score for each of the buckets the wire goes through is used to compute the spacing for the wire segment in the bucket. To calculate the congestion score for a bucket, it is important to remember that design rules require a certain amount of free space on each side of a routing wire. Thus, a layer having spaces to accommodate ten routing wires which has five routed therethrough (wire-space-wire-space . . .) may have 100% congestion because the layer cannot accommodate any more routine wires while maintaining the separation design rule.

Given the above, one can see that a layer having 50% congestion can roughly be thought of as having a number of routing wires with one empty space on one side and two empty spaces on the other—omitting the required space on each side of the wire, it has one wire for two usable spaces). A layer having 33% congestion can similarly be thought of as having routing wires with one space on one side and three spaces on the other, or with two spaces on one side and two spaces on the other (one wire in three usable spaces). A layer having 25% congestion can be thought of as having routing

wires with two spaces on one side and three spaces on the other (one wire in four usable spaces). Further discrete points can be derived with more sparse routing layers.

Given a maximum of three spaces between adjacent wires in a routing layer, a suitable equivalence chart might be:

Congestion Score	Corresponds to
0–23%	3 spaces/3 spaces
23–29%	2 spaces/3 spaces
29–42%	2 spaces/2 spaces or 1 space/3 spaces
42–75%	1 space/2 spaces
75–100%	1 space/1 space

In a variation on the preferred embodiment the spacing value for the wires are rounded to the nearest integers and a congestion score of more than 75% is considered to have a spacing of 1 on both sides (the minimum spacing which could conform to the design rules), a congestion score of 50% represents a spacing of 1 on one side and 2 on the other side and a score below 50% uses spacing of 2 on both sides. Hence if the initial number of routable spaces available in a bucket is 10 and we have routed three wires through it then the congestion score for the bucket is 30%. Each wire running through this bucket will be assumed to have a spacing of 2 on both sides.

The interconnect wire capacitance determination makes use of the spacing thus computed to compute the lateral capacitance for the wire segment. As is known in the art, this can be done by taking a number of parameters of the technology into account; for example, design rule spacings, distance of the routing layer from the substrate, thickness and material of the wires and the like. Generally speaking, the most important factors are the width and spacing of the wires.

The capacitance of the whole net is then computed by adding up the contribution of each wire segment belonging to the net. It may then be used where needed in optimization operations, e.g., inserting repeaters, choosing appropriate strengths, etc.

Based on the routed cell arrangement resulting from the initial Steiner tree arrangement, the cell areas can be recalculated in Step **355** using Equation 15 with the new wire loads substituted therein. At this point, the cell placement will likely be somewhat unbalanced. This imbalance may take several forms:

- widely varying cell utilization percentages—for example, if the core utilization before the cell area recalculation is 90%, after recalculation some buckets **410** will have higher utilization percentages and some buckets **410** will have lower utilization percentages. This is undesirable because, for example, overutilized buckets **410** may present obstacles to wire routing or usage of pads.
- cell recalculation enlarges the size of some cells **420** so that they do not fit within their buckets **410**, or so that they overlap other cells **420**.
- cell recalculation results in too much wasted area, i.e., unutilized core area.

To correct these problems, an iterative procedure is used. First, the current layout is checked to see if it meets given utilization constraints such as core utilization percentage in Step **360**. If so, the placement procedure is complete and this part of the routine ends. If not, i.e., if the total area A_{total} of the cells **420** does not fit in the core **400** within the given predetermined utilization constraints, the procedure returns to Step **345** where repartitioning is conducted by coarse placement based on the last-determined cell areas from Step **350**, and the repartition-recalculation-checking loop is iteratively executed again based on the newly-calculated cell areas and wire loads to further converge toward an acceptable placement.

Additional analysis shows that it is always possible to find a floorplan where the total area of the cells **420** matches the core area. Consider a coarse placement where each cell **420** has a location (xi, yi) and an area based on the load it drives as outlined above. From Equation 14 above, the total area of the design is

$$A_{Cell}=K_1+WL \tag{22}$$

Now, assume both x and y directions are stretched by a factor of α. The length of each wire is increased by α, and since the cell area is linearly dependent on the wire length,

$$A_{Scaled\ Cell}=\alpha(K_1+WL) \tag{23}$$

However, by scaling the core **400** by a factor of a its area will increase quadratically:

$$A_{Scaled\ Core}=\alpha^2 A_{Core} \tag{24}$$

Since the core area increases more rapidly that the cell area as they are scaled, at some point the core area will be equal to and then exceed the cell area. This point can be found by setting the scaled core area equal to the total scaled cell area and solving for α:

$$\alpha^2 A_{Core}=\alpha(K_1+WL) \tag{25}$$

$$\alpha = \frac{B+WL}{A_{Core}} \tag{26}$$

This is the factor by which the core **400** must be enlarged to accommodate the total cell area.

After a satisfactory placement has been found in Step **560**, the cell area in individual buckets **410** is balanced to balance routing resource usage and area usage among all buckets **410**. First, a global router assigns routes to all nets in Step **365** and an analysis of routing resources on the core **400** determines congested areas. In Step **370**, cells **420** in the most congested areas are "padded" by arbitrarily increasing their areas slightly, and cells **420** in the most underutilized areas are "shrunk" by arbitrarily reducing their areas slightly. This tends to increase the rate at which cells **420** migrate from overutilized areas to underutilized areas.

Next, a bucket equalization process is applied to the cells **420** to move cells **420** from overutilized buckets **410** to underutilized ones in Step **375**. This is a sort of "bucket brigade" movement in which a cell **420** moves at most from one bucket **410** to an adjacent bucket **410**. For example, in a series of ten consecutively numbered buckets **410** on a horizontal path, if cells **420** need to be moved from bucket **1** to bucket **10**, some cells **420** are moved from bucket **1** to bucket **2**; some from bucket **2** to bucket **3**, etc. As cells **420** move from one bucket **410** to another, the loads of nets attached to them change. This causes a corresponding change in the area of other cells **420** in the design, and these are corrected locally rather than through a global recalculation process. To ensure that changes to cell areas are minimized, cell movements along many different paths are examined and only the best used.

Finally, in the pairwise refinement process of Step **380**, a mincut process is applied between adjacent buckets **410** in

a sweeping fashion as shown in FIG. **8**. Starting from the topmost corner of core **400**, each bucket **410** and its immediate neighbors to the right and bottom are repartitioned in order to reduce the number of crossing nets. One full pass of the repartitioning ends when the bottom rightmost bucket **410** is reached. At this point, the total wire length in the circuit is computed in Step **385** and the areas of all cells **420** are readjusted in Step **390**. If there is an improvement in wire length, another iteration through the process is begun at Step **365**; if not, this part of the process is complete.

After the strengths of the cells **420** are optimized based on the timing analysis using the global routing topologies, track routing is performed which assigns a specific routing space in the bucket **410** to each wire. The final stage of physical routing software puts the physical wires following the space assignment made by the track routing as closely as possible and completes the connections for all the cells.

The present invention has been described above in connection with a preferred embodiment thereof; however, this has been done for purposes of illustration only, and the invention is not so limited. Indeed, variations of the invention will be readily apparent to those skilled in the art and also fall within the scope of the invention.

For example, the embodiment has been described in connection with a discrete integer spacing architecture; however, the invention is not so limited and may be used with fractional or other non-integer systems as well. Further, although in the preferred embodiment a maximum spacing of three was used, higher-order spacings may be used as well.

What is claimed is:

1. A method of estimating capacitance of interconnect wires in an integrated circuit chip design, the method comprising:

grouping a plurality of cells in the design into a plurality of buckets;

maintaining a congestion score for each bucket;

when routing a wire through a bucket, modifying the congestion score accordingly; and

using the congestion score to calculate an estimated capacitance for the wire.

2. The method of claim **1**, wherein the congestion score is a ratio of a number of available wire routing spaces in a given layer of the bucket in a given direction to a total number of wire routing spaces in the given layer of the bucket in the given direction.

3. The method of claim **1**, wherein maintaining the congestion score for each bucket includes excluding a wire routing space required to be empty in order to meet design rules from consideration as an available wire routing space.

4. The method of claim **1**, wherein using the congestion score to calculate an estimated capacitance for the wire includes estimating a spacing on at least one side of wires in a bucket based on that bucket's congestion score.

5. The method of claim **4**, wherein using the congestion score to calculate an estimated capacitance for the wire includes estimating a spacing on both sides of wires in the bucket based on that bucket's congestion score.

6. The method of claim **4**, wherein a range of congestion scores is equivalent to a given spacing configuration for wires in the bucket.

7. The method of claim **1**, wherein using the congestion score to calculate an estimated capacitance for the wire includes adding together a plurality of estimated capacitances for segments of the wire in different buckets.

8. The method of claim **1**, wherein the calculated estimated capacitance includes a lateral capacitance component.

* * * * *

TAB 9

IW 1491802

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office

August 03, 2006

THIS IS TO CERTIFY THAT ANNEXED IS A TRUE COPY FROM THE RECORDS OF THIS OFFICE OF THE FILE WRAPPER AND CONTENTS OF:

APPLICATION NUMBER: *09/579,966*
FILING DATE: *May 26, 2000*
PATENT NUMBER: *6,519,745*
ISSUE DATE: *February 11, 2003*

By Authority of the
Under Secretary of Commerce for Intellectual Property
and Director of the United States Patent and Trademark Office

P. SWAIN
Certifying Officer

#6/Response w/Exhibit A-tocl 7/16/02

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re Application of: **Srinivas, et al.** Serial No. **09/579,966** Filed: **May 26, 2000** For: **System And Method For Estimating Capacitance of Wires Based on Congestion Information**	Examiner: H. Rossoshek Art Unit: 2825 Atty. Docket No. 054355/0269890 Client Ref: MAG-024 **CERTIFICATE OF MAILING** *I certify that the enclosed papers are being deposited with the U.S. Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, Washington, D.C. 20231, on **June 14, 2002*** Maria English Printed Name: Maria English

RESPONSE TO OFFICE ACTION AND PETITION FOR EXTENSION OF TIME

Assistant Commissioner of Patents
Washington, DC 20231
BOX: Amendment

Sir:

This communication is in response to the non-final Office Action mailed on February 14, 2002 for the above-identified patent application. Applicants hereby petition for a one-month extension of time extending the period for response from May 14, 2002 to June 14, 2002, and enclose the requisite fee herewith. Please enter the below remarks and amend the application as follows.

Remarks

No claims have been added or cancelled by this amendment. Accordingly, claims 1-8 remain pending in the application.

Objections to the specification

The disclosure was objected to due to the assertion that there are no references to Figures 1 and 9. With regard to Figure 1, please note that on page 1, line 16, specific reference is made to "FIG. 1" with the text on the three pages following that reference describing the contents of Figure 1. With respect to Figure 9, please note that on page 5, line 4, specific reference is made to Figure 9, with the text in the paragraph following that reference describing the contents of Figure 9. Applicants therefore respectfully submit that this correction is not required and that the objection be withdrawn.

Objections to the drawings

In response to the clarity objection to Figure 7, Applicants hereby submit a new copy (enclosed) of Figure 7 which is more clear. Applicants therefore request that this objection be withdrawn.

Rejections under 35 USC 102

Claims 1, 7 and 8 were rejected under 35 USC 102(e) as being anticipated by Darden et al. (U.S. Patent Number 6,185,272) ("Darden"). Applicants respectfully disagree.

In brief, the invention is directed toward a method of estimating capacitance of interconnect wires in an integrated circuit chip design. The method is based upon maintaining and modifying a congestion score for each bucket into which the design is split. The congestion score of a bucket is then used to calculate an estimated capacitance for a wire which is routed into that bucket. Specifically, claim 1 recites:

> "A method of estimating capacitance of interconnect wires in an integrated circuit chip design, the method comprising:
> grouping a plurality of cells in the design into a plurality of buckets;
> *maintaining a congestion score* for each bucket;
> when routing a wire through a bucket, *modifying the congestion score* accordingly; and
> *using the congestion score to calculate an estimated capacitance* for the wire"

The Office Action cites Darden as anticipating claim 1. However, the claim elements italicized above are nowhere taught or disclosed in Darden. While Darden does disclose organizing physical chip data into 2D buckets called grid cells, these grids are used merely to define at grid intersections the type of chip element neighboring the cell. Thus, Darden cannot teach that such grid cells are used for maintaining, modifying and using a congestion score as does claim 1 of Applicants' invention.

The Office Action cites column 14, lines 38-42 of Darden as teaching "maintaining a congestion score for each bucket". However, Darden actually teaches away from such a step. The cited portion of Darden discloses instead searching for elements that surround a given grid unit, and does not relate to the elements within the grid unit. Further more, the concept of a congestion score is not discussed. Instead of "maintaining a congestion score for each bucket", as does Applicants' invention, Darden uses a grid structure to first search for neighboring elements outside a given grid that may affect that grid.

Likewise, since Darden nowhere teaches or discloses the concept of a congestion score, it cannot teach "modifying the congestion score" as also set forth in claim 1. The Office Action cites column 4, lines 5-19 as teaching the "modifying the congestion score" element of Applicants' claim 1. Rather, the cited reference teaches how searches are performed in elements neighboring a particular wire and not "modifying the congestion score."

Finally, the Office Action cites column 14, lines 43-45 as teaching "using the congestion score to calculate an estimated capacitance for the wire". Again, since a congestion score is not mentioned, such a score could not be used to calculate a capacitance value. Instead of calculate a capacitance value, Darden "selects" a capacitance value from table look-up based upon a search of neighboring elements.

Since Darden does not disclose the elements of claim 1, Darden fails to anticipate claim 1. Therefore, Applicants respectfully submit claim 1 is allowable and respectfully requests that the rejection be withdrawn.

Since claims 7 and 8 depend upon claim 1, which Applicants have demonstrated is allowable, these claims are also allowable. Applicants therefore respectfully request that all the rejections under 35 USC 102 be withdrawn.

Rejections under 35 USC 103

Claim 2 is rejected under 35 USC 102(e) as being anticipated by Darden et al. (U.S. Patent Number 6,185,272) ("Darden") in view of Kuroda et al (U.S. Patent Number 6,311,139) ("Kuroda"). Applicants respectfully disagree.

Kuroda relates to determining the yield of an integrated circuit device by estimating the density of defects therein and using that defect density to compute an estimated yield. Kuroda does not relate to wire capacitances at all. Therefore, there is no motivation to combine Darden and Kuroda.

Assuming such a motivation solely to illustrate the inappropriateness of the rejection, the combination would still fail to teach or disclose claim 2. Claim 2 recites that the congestion score "is a

ratio of *a number of available* wire routing *spaces* in a given layer of the bucket in a given direction to *a total number* of wire routing *spaces* in the given layer of the bucket in the given direction". The Office Action cites Kuroda, column 7, lines 55-67, column 12, lines 44-45 and column 14, lines 29-34, as teaching the congestion score being "a ratio of a number of available wire routing spaces in a given layer of the bucket in a given direction to a total number of wire routing spaces in the given layer of the bucket in the given direction". However, the cited portions of Kuroda discuss only the ratio of transistor density to mean transistor density, and not spaces in between those transistors. The congestion score measures the ratio of available spaces to the total spaces available for wiring. Kuroda does not disclose the ratio of spacing, but rather the ratio of the actual elements, i.e. transistors, that are present. Thus, even if Kuroda and Darden were combined, the combination would still fail to yield claim 2, particularly since Darden makes no mention of a congestion score and the ratio in Kuroda is entirely different from the congestion score defined in claim 2.

Applicants therefore respectfully submit that claim 2 is allowable and request that the rejection under 35 USC 103 be withdrawn.

Applicants respectfully submit that all claims are thus in condition for allowance and respectfully requests a notice to that effect at the earliest.

Respectfully submitted,

By _____
Anand Sethuraman, Reg. No. 43,351

PILLSBURY WINTHROP LLP
1600 Tysons Blvd.
McLean, VA 22012
650-233-4790
650-233-4545 (fax)

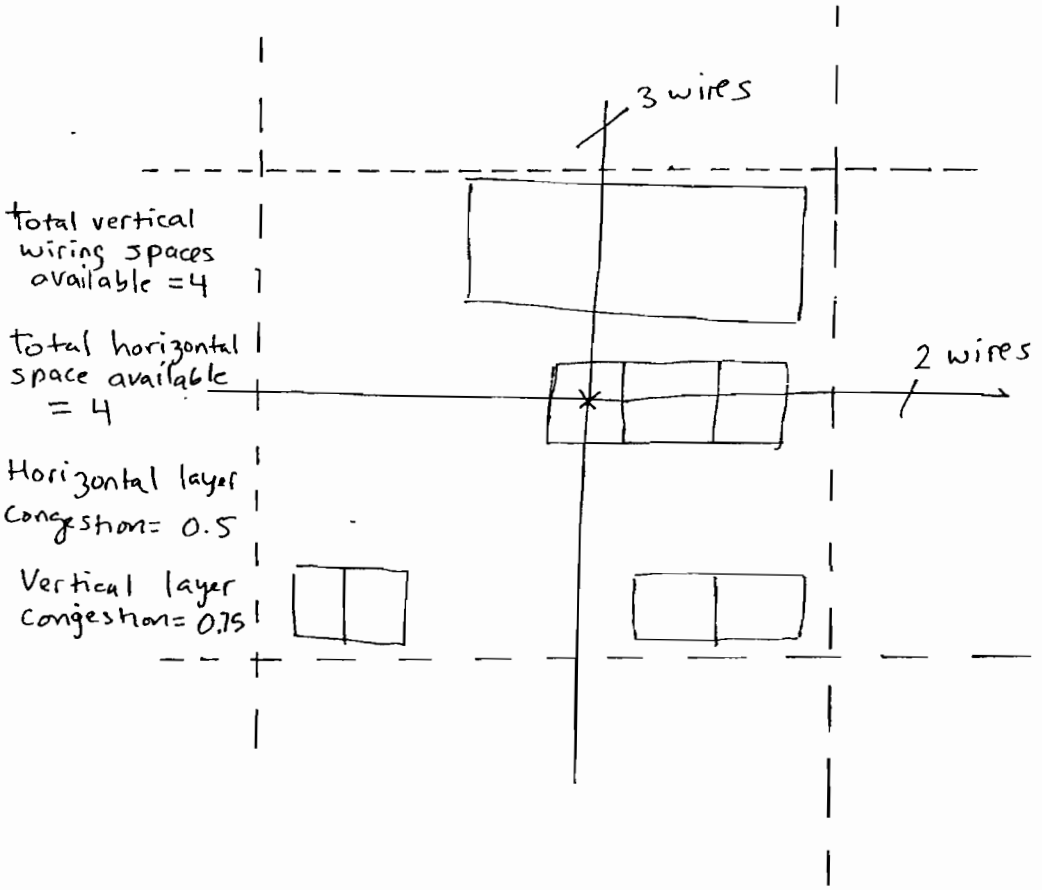

3 wires
total vertical wiring spaces available = 4
total horizontal space available = 4
2 wires
Horizontal layer congestion = 0.5
Vertical layer congestion = 0.75

Figure 7

TAB 10

The

AMERICAN HERITAGE®

College

dic·tion·ar·y

FOURTH EDITION

The American Heritage® College Dictionary

FOURTH EDITION

Visit our website: www.houghtonmifflinbooks.com

The American Heritage college dictionary.-- 4th ed.
p.cm.
Based on the fourth ed. of the American Heritage dictionary.
ISBN 0-618-09848-8 (thumb edge) --
ISBN 0-618-19604-8 (deluxe binding)
1. English language--Dictionaries. 2. Americanisms. I Houghton Mifflin Company. II. American Heritage dictionary.

PE1628 .A6227 2002
423--dc21

2001039826

Manufactured in the United States of America

a mass, as sores in a rash. ❖ *n.* **1.** One of two or more confluent streams. **2.** A tributary. [ME < Lat. *cōnfluēns, cōnfluent-*, pr. part. of *cōnfluere*, to flow together : *com-*, com- + *fluere*, to flow.]

con•flux (kŏn′flŭks′) *n.* A confluence. [< Lat. *cōnflūxus*, p. part. of *cōnfluere*, to flow together. See CONFLUENT.]

con•fo•cal (kŏn-fō′kəl) *adj.* Having the same focus or foci. Used of a lens. **—con•fo′cal•ly** *adv.*

con•form (kən-fôrm′) *v.* **-formed, -form•ing, -forms** *—intr.* **1.** To correspond in form or character; be similar. **2.** To act or be in accord or agreement; comply. **3.** To act in accordance with current customs or modes. See Syns at **adapt.** *—tr.* To bring into agreement or correspondence; make similar. [Ult. < Lat. *cōnfōrmāre*, to shape after : *com-*, com- + *fōrmāre*, to shape (< *fōrma*, shape; see FORM).] **—con•form′er** *n.*

con•form•a•ble (kən-fôr′mə-bəl) *adj.* **1.** Corresponding; similar. **2.** Quick to comply; submissive. **3.** *Geology* Of, relating to, or being strata that are parallel to each other without interruption. **—con•form′a•bil′i•ty, con•form′a•ble•ness** *n.* **—con•form′a•bly** *adv.*

con•for•mal (kən-fôr′məl) *adj.* **1.** *Mathematics* Of, relating to, or being a mapping in which all angles between intersecting curves remain unchanged. **2.** Of or relating to a map projection in which small areas are rendered with true shape. [LLat. *cōnfōrmālis*, similar : Lat. *com-*, com- + Lat. *fōrma*, shape; see FORM.]

con•for•mance (kən-fôr′məns) *n.* Conformity.

con•for•ma•tion (kŏn′fər-mā′shən) *n.* **1.** The act of conforming or the state of being conformed. **2.** The structure or outline of an item or entity, determined by the arrangement of its parts. **3.** A symmetrical arrangement of the parts of a thing. **4.** A spatial arrangement of atoms in a molecule brought about by free rotation of the atoms about a single chemical bond. **—con′for•ma′tion•al** *adj.* **—con′for•ma′tion•al•ly** *adv.*

con•form•ist (kən-fôr′mĭst) *n.* A person who uncritically or habitually conforms to the customs, rules, or styles of a group. ❖ *adj.* Marked by conformity or convention. **—con•form′ism** *n.*

con•form•i•ty (kən-fôr′mĭ-tē) *n., pl.* **-ties 1.** Similarity in form or character; agreement. **2.** Action or behavior in correspondence with current customs, rules, or styles. **3.** The relationship between adjacent rock strata representing a continuous period of time.

con•found (kən-found′, kŏn-) *tr.v.* **-found•ed, -found•ing, -founds 1.** To cause to become confused or perplexed. **2.** To fail to distinguish; mix up. **3.** To make (something bad) worse. **4.** To cause to be ashamed; abash. **5.** To damn. **6a.** To frustrate. **b.** *Archaic* To bring to ruination. [ME *confounden* < AN *confundre* < Lat. *cōnfundere*, to mix together, confuse : *com-*, com- + *fundere*, to pour; see **gheu-** in App.] **—con•found′er** *n.*

con•found•ed (kən-foun′dĭd, kŏn-) *adj.* **1.** Confused; befuddled. **2.** Used as an intensive: *a confounded fool.* **—con•found′ed•ly** *adv.* **—con•found′ed•ness** *n.*

con•fra•ter•ni•ty (kŏn′frə-tûr′nĭ-tē) *n., pl.* **-ties** An association of persons united in a common purpose or profession. [ME *confraternite* < OFr. < Med.Lat. *cōnfrāternitās* < *cōnfrāter*, colleague. See CONFRERE.]

con•frere (kŏn′frâr′) *n.* A fellow member of a fraternity or profession; a colleague. [ME < OFr. < Med.Lat. *cōnfrāter* : Lat. *com-*, com- + Lat. *frāter*, brother; see **bhrāter-** in App.]

con•front (kən-frŭnt′) *v.* **-front•ed, -front•ing, -fronts** *—tr.* **1.** To come face to face with, esp. with defiance or hostility: *confronted her accuser in a court of law.* **2.** To bring face to face with: *was confronted with the truth.* **3.** To come up against; encounter: *confronted danger.* *—intr.* To engage in confrontation. [Fr. *confronter* < OFr., to adjoin < Med.Lat. *cōnfrontāre* : Lat. *com-*, com- + Lat. *frōns, front-*, front.] **—con′fron•ta′tive** *adj.* **—con•front′er** *n.* **—con•front′ment** *n.*

con•fron•ta•tion (kŏn′frŭn-tā′shən) *n.* **1.** The act of confronting or the state of being confronted, esp. a meeting face to face. **2a.** A conflict involving armed forces: *a nuclear confrontation.* **b.** Discord or a clash of opinions and ideas: *ideological confrontation.* **3.** A focused comparison. **—con′fron•ta′tion•al** *adj.* **—con′fron•ta′tion•ist** *n.*

Con•fu•cius (kən-fyōō′shəs) Orig. Kong Fuzi. c. 551–479 B.C. Chinese philosopher whose *Analects* contain a collection of his sayings. **—Con•fu′cian** (-shən) *adj. & n.* **—Con•fu′cian•ism** *n.* **—Con•fu′cian•ist** *n.*

con•fuse (kən-fyōōz′) *v.* **-fused, -fus•ing, -fus•es** *—tr.* **1a.** To cause to be unable to think with clarity or act with intelligence or understanding. **b.** To cause to feel embarrassment. **2a.** To fail to differentiate from another: *confused flattery with admiration.* **b.** To make opaque; blur. **c.** To assemble without order or sense; jumble. **3.** *Archaic* To bring to ruination. *—intr.* To make something unclear or incomprehensible. [ME *confusen* < OFr. *confus*, perplexed < Lat. *cōnfūsus*, p. part. of *cōnfundere*, to mix together. See CONFOUND.] **—con•fus′a•ble** *adj.* **—con•fus′ing•ly** *adv.*

SYNONYMS *confuse, addle, befuddle, discombobulate, fuddle, muddle, throw* These verbs mean to cause to be unclear in mind or intent: *heavy traffic that confused the driver; problems that addle my brain; a question that befuddled even the professor; was discombobulated by the possibilities; a plot line that fuddled my comprehension; a student who was muddled by facts and figures; behavior that really threw me.*

con•fused (kən-fyōōzd′) *adj.* **1.** Being unable to think with clarity or act with understanding and intelligence. **2a.** Lacking logical order or sense. **b.** Chaotic; jumbled. **—con•fus′ed•ly** (-fyōō′zĭd-lē) *adv.* **—con•fus′ed•ness** *n.*

con•fu•sion (kən-fyōō′zhən) *n.* **1a.** The act of confusing or the state of being confused. **b.** An instance of being confused. **2.** *Psychology* Impaired orientation with respect to time, place, or person; a disturbed mental state. **—con•fu′sion•al** *adj.*

con•fu•ta•tion (kŏn′fyōō-tā′shən) *n.* **1.** The act of confuting. **2.** Something that confutes.

con•fute (kən-fyōōt′) *tr.v.* **-fut•ed, -fut•ing, -futes 1.** To prove to be wrong or in error; refute decisively. **2.** *Obsolete* To confound. [Lat. *cōnfūtāre.*] **—con•fut′a•ble, con•fu′ta•tive** (kən-fyōō′tə-tĭv) *adj.* **—con•fut′er** *n.*

cong. *abbr. Pharmacology* congius (gallon)

Cong. *abbr.* **1.** Congregational **2.** Congress

con•ga (kŏng′gə) *n.* **1.** A dance in which the dancers form a long winding line. **2.** Music for this dance. **3.** A conga drum. ❖ *intr.v.* **-gaed, -ga•ing, -gas** To perform this dance. [Am.Sp. *(danza) Conga*, Congo "(dance) < Sp. *Congo*, of the Congo < Kongo *-kongo*, Kongo language and people.]

conga drum *n.* A tall, usu. tapering single-headed drum typically played by beating with the hands.

con game *n. Slang* A confidence game.

con•gé (kŏn′zhā′, -jā′, kôN-zhā′) also **con•gee** (kŏn′jē) *n.* **1.** Formal or authoritative permission to depart. **2.** An abrupt dismissal. **3.** A leave-taking. **4.** A formal bow. **5.** *Architecture* A concave molding. [ME *conge* and Fr. *congé*, both < OFr. *congie* < Lat. *commeātus* < p. part. of *commeāre*, to come and go : *com-*, com- + *meāre*, to go.]

con•geal (kən-jēl′) *v.* **-gealed, -geal•ing, -geals** *—intr.* **1.** To solidify by or as if by freezing. **2.** To coagulate; jell. *—tr.* To cause to congeal. [ME *congelen* < OFr. *congeler* < Lat. *congelāre* : *com-*, com- + *gelāre*, to freeze; see **gel-** in App.] **—con•geal′a•ble** *adj.* **—con•geal′er** *n.* **—con•geal′ment** *n.*

con•gealed salad (kən-jēld′) *n. Chiefly Southern US* A molded salad made of flavored gelatin, chopped fruits or vegetables, and sometimes other ingredients.

con•ge•la•tion (kŏn′jə-lā′shən) *n.* The process of congealing or the state of being congealed.

con•ge•ner (kŏn′jə-nər) *n.* **1.** A member of the same kind, class, or group. **2.** An organism belonging to the same taxonomic genus as another organism. [< Lat., of the same race : *com-*, com- + *genus, gener-*, race; see **genə-** in App.] **—con′ge•ner′ic** (-nĕr′ĭk), **con•gen′er•ous** (kən-jĕn′ər-əs, kŏn-) *adj.*

con•gen•ial (kən-jēn′yəl) *adj.* **1.** Having the same tastes, habits, or temperament; sympathetic. **2.** Of a pleasant disposition; friendly: *a congenial host.* **3.** Suited to one's needs or nature; agreeable. [Prob. < CON- + Lat. *genius*, the personification of one's natural inclinations; see GENIUS.] **—con•ge′ni•al′i•ty** (-jē′nē-ăl′ĭ-tē), **con•gen′ial•ness** *n.* **—con•gen′ial•ly** *adv.*

con•gen•i•tal (kən-jĕn′ĭ-tl) *adj.* **1.** Existing at or before birth, as a defect or medical condition. **2.** Being or having an essential characteristic as if by nature; inherent or inveterate. [< Lat. *congenitus* : *com-*, com- + *genitus*, born, p. part. of *gignere*, to bear; see **genə-** in App.] **—con•gen′i•tal•ly** *adv.*

congenital anomaly *n.* See **birth defect.**

con•ger (kŏng′gər) *n.* Any of various large scaleless marine eels of the family Congridae, esp. *Conger oceanicus*, native to Atlantic waters. [ME *congre* < OFr., prob. < LLat. *congrus* < Lat. *conger* < Gk. *gongros.*]

con•ge•ries (kən-jîr′ēz′, kŏn′jə-rēz′) *n.* (*used with a sing. verb*) A collection; an aggregation. [Lat. *congeriēs* < *congerere*, to heap up. See CONGEST.]

con•gest (kən-jĕst′) *v.* **-gest•ed, -gest•ing, -gests** *—tr.* **1.** To overfill or overcrowd. **2.** *Pathology* To cause the accumulation of excessive blood or fluid in (a vessel or organ). *—intr.* To become congested. [Lat. *congerere, congest-*, to heap up, crowd together : *com-*, com- + *gerere*, to carry.] **—con•ges′tion** *n.* **—con•ges′tive** *adj.*

congestive heart failure *n.* A condition marked by weakness, edema, and shortness of breath that is caused by inadequate blood circulation in the peripheral tissues and the lungs.

con•gi•us (kŏn′jē-əs) *n., pl.* **-gi•i** (-jē-ī′) **1.** *Pharmacology* A gallon. **2.** An ancient Roman measure for liquids, equal to about seven eighths of a US gallon (3.3 liters). [ME, a liquid measure < Lat. < Gk. *konkhion*, dim. of *konkhē, konkhos*, shellful.]

con•glo•bate (kŏn-glō′bāt′, kŏng′glō-) *tr.v.* **-bat•ed, -bat•ing, -bates** To form into a globe or ball. [Lat. *conglobāre, conglobāt-* : *com-*, com- + *globus*, ball.] **—con•glo′bate** *adj.* **—con′glo•ba′tion** *n.*

con•globe (kən-glōb′) *tr.v.* **-globed, -glob•ing, -globes** To conglobate.

con•glom•er•ate (kən-glŏm′ə-rāt′) *intr. & tr.v.* **-at•ed, -at•ing, -ates** To form or cause to form into an adhering or rounded mass. ❖ *n.* (-ər-ĭt) **1.** A corporation made up of a number of different companies that operate in diversified fields. **2.** A collected heterogeneous mass; a cluster: *a conglomerate of colors.* **3.** *Geology* A rock consisting of pebbles and gravel embedded in cement. ❖ *adj.* (-ər-ĭt) **1.** Gathered into a mass; clustered. **2.** *Geology* Made

Confucius

ă	pat	oi	boy
ā	pay	ou	out
âr	care	o͝o	took
ä	father	o͞o	boot
ĕ	pet	ŭ	cut
ē	be	ûr	urge
ĭ	pit	th	thin
ī	pie	*th*	this
îr	pier	hw	which
ŏ	pot	zh	vision
ō	toe	ə	about, item
ô	paw		

Stress marks:
ˈ (primary);
ˌ (secondary), as in **lexicon** (lĕkˈsĭ-kŏnˌ)

TAB 11

CONFIDENTIAL EXHIBIT

TAB 12

CONFIDENTIAL EXHIBIT

TAB 13

CONFIDENTIAL EXHIBIT

TAB 14

CONFIDENTIAL EXHIBIT

TAB 15

US006230304B1

(12) **United States Patent**
Groeneveld et al.

(10) **Patent No.: US 6,230,304 B1**
(45) **Date of Patent: May 8, 2001**

(54) **METHOD OF DESIGNING A CONSTRAINT-DRIVEN INTEGRATED CIRCUIT LAYOUT**

(75) Inventors: **Patrick R. Groeneveld; Lukas P. P. P. van Ginneken**, both of San Jose, CA (US)

(73) Assignee: **Magma Design Automation, Inc.**, Cupertino, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/054,319**

(22) Filed: **Apr. 2, 1998**

Related U.S. Application Data

(60) Provisional application No. 60/068,827, filed on Dec. 24, 1997.

(51) **Int. Cl.**[7] **G06F 17/50**

(52) **U.S. Cl.** **716/7**

(58) **Field of Search** 716/9, 18

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,484,292 11/1984 Hong et al. .
4,615,011 9/1986 Linsker .

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

0677813 10/1995 (EP) G06F/17/50

OTHER PUBLICATIONS

M. Pedram, "Logical–Physical Co–Design for Deep Submicron Circuits: Challenges and Solutions", Proceedings of the ASP–DAC '98 Asian and S. Pacific, Design Automation Conference 1998, pp. 137–142.

T. Xue et al., "Post Global Routing Crosstalk Synthesis", IEEE, vol. 16, No. 12, Dec. 1997, pp. 1418–1430.

T. Koide et al., "A Timing–Driven Global Routing Alogrithm With Pin Assignment, Block Reshaping, and Positioning for Building Block Layout", Proceedings of the ASP–DAC '98 Asian and S. Pacific, Design Automation Conference, 1998, pp. 577–583.

S–Y Oh et al., "A Scaling Scheme and Optimization Methodology for Deep Sub–Micron Interconnect", Proceedings of the International Conference on Computer Design, ICC VLSI, Oct. 1996, pp. 320–325.

Groeneveld, Patrick: "Context–driven Channel Routing," Ph.D. Dissertation, Delft University of Technology, Delft, The Netherlands, Jun. 1991.

Internet Publication http://www.isdmag.com/Editorial/1997/Editorial9704.html.

Internet Publication http://www.isdmag.com/EEdesign/FocusReport9706.html.

Internet Publication http://virtual.union.edu/~templej/lab3/index.html.

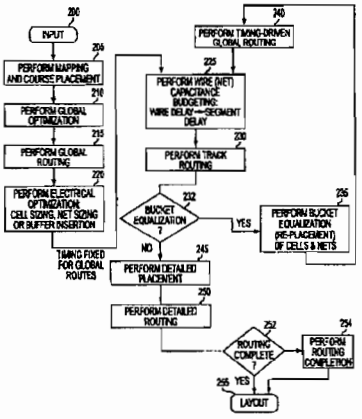
(List continued on next page.)

Primary Examiner—Matthew Smith
Assistant Examiner—Thuan Do
(74) *Attorney, Agent, or Firm*—Pillsbury Winthrop LLP

(57) **ABSTRACT**

An automated method for designing an integrated circuit layout with a computer, based upon an electronic circuit description and upon a selected plurality of cells from a cell library, comprising the steps of: (a) assigning each of the cells to one of a plurality of buckets designated on the integrated circuit layout, each of the cells being connected to one of the other cells; (b) performing global routing to connect at least some of the selected cells of step (a) together such that global routes are formed to provide net topology information; (c) performing track routing which sets the position of each of the global routes; (d) performing detailed placement such that the positions of all selected cells are fixed within each of the buckets designated on the integrated circuit layout; and (e) performing detailed routing such that detailed routes are formed to complete the integrated circuit layout.

86 Claims, 21 Drawing Sheets



U.S. PATENT DOCUMENTS

4,949,275	8/1990	Nonaka .	
5,046,017	9/1991	Yuyama et al.	364/491
5,047,949	9/1991	Yamaguchi et al. .	
5,065,355	11/1991	Hayase .	
5,072,402	12/1991	Ashtaputre et al. .	
5,191,542	3/1993	Murofushi	364/491
5,245,550	9/1993	Miki et al. .	
5,249,134	9/1993	Oka .	
5,341,310	8/1994	Gould et al. .	
5,361,214	11/1994	Aoki .	
5,394,337	2/1995	Shinjo .	
5,404,312	4/1995	Tawada .	
5,452,239	9/1995	Dai et al. .	
5,461,576	10/1995	Tsay et al.	364/490
5,471,398	11/1995	Stephens .	
5,483,461	1/1996	Lee et al. .	
5,485,396	1/1996	Brasen et al. .	
5,537,330	7/1996	Damiano et al. .	
5,590,050	12/1996	Okuda .	
5,598,347	1/1997	Iwasaki .	
5,625,568	4/1997	Edwards et al. .	
5,629,860	5/1997	Jones et al.	364/490
5,633,806	5/1997	Yusa et al. .	
5,636,129	6/1997	Her .	
5,638,288	6/1997	Deeley .	
5,649,170	7/1997	Chappell et al.	395/500
5,654,898	8/1997	Roetcisoender et al. .	
5,657,241	8/1997	Butts et al. .	
5,657,242	8/1997	Sekiyama et al. .	
5,663,892	9/1997	Hayashi et al. .	
5,666,288	9/1997	Jones et al. .	
5,675,501	10/1997	Aoki .	
5,682,323	10/1997	Pasch et al. .	
5,689,433	11/1997	Edwards .	
5,715,171	2/1998	Mori et al. .	
5,729,469	3/1998	Kawakami .	
5,737,237	4/1998	Tanaka et al. .	
5,745,373	4/1998	Watai et al. .	
5,757,658	5/1998	Rodman et al. .	
5,761,078	6/1998	Fuller et al. .	
5,764,525	6/1998	Mahmood et al. .	
5,764,530	6/1998	Yokomaku .	
5,764,532	6/1998	Patel .	
5,764,534	6/1998	Goetting .	
5,768,146	6/1998	Jassawski .	
5,774,371	6/1998	Kawakami .	
5,778,216	7/1998	Venkatesh .	
5,838,583 *	11/1998	Varadarajan	716/9
6,102,964 *	8/2000	Tse	716/18

OTHER PUBLICATIONS

Internet Publication http://virtual.edu/~temple/lab3/k_maps.html.

Internet Publication http://virtual.union.edu/~templej/lab3/counter.html.

Internet Publication http://virtual.union.edu/~templej/lab3/counter_ledit.html.

Internet Publication http://virtual.union.edu/~templej/lab3/counter_wave.html.

Internet Publication http://www.cadence.com/features/DSM.html.

Internet Publication http://www.isdmag.com/Editorial/1997/EDAambitFeature9706.html.

* cited by examiner

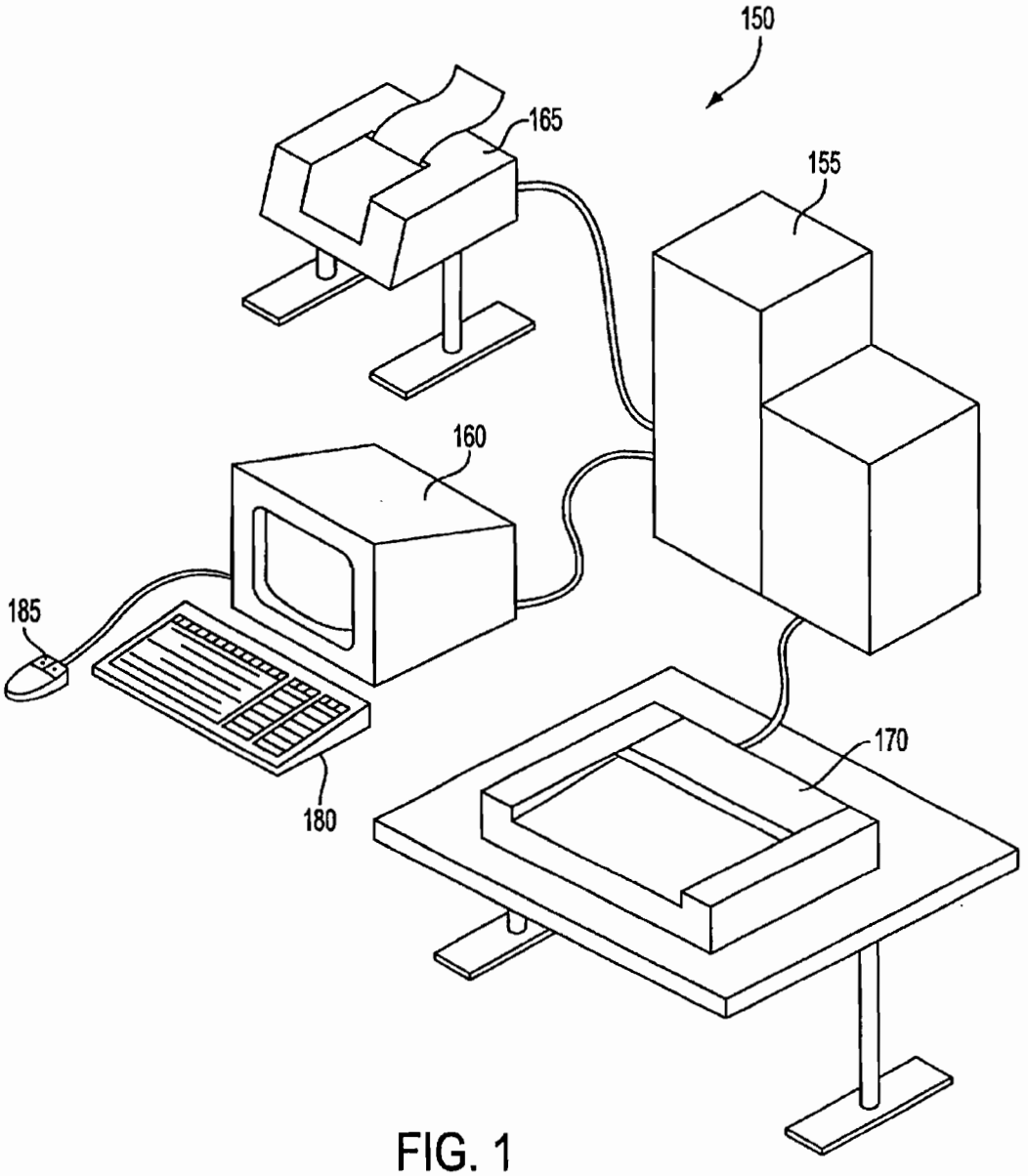


FIG. 1

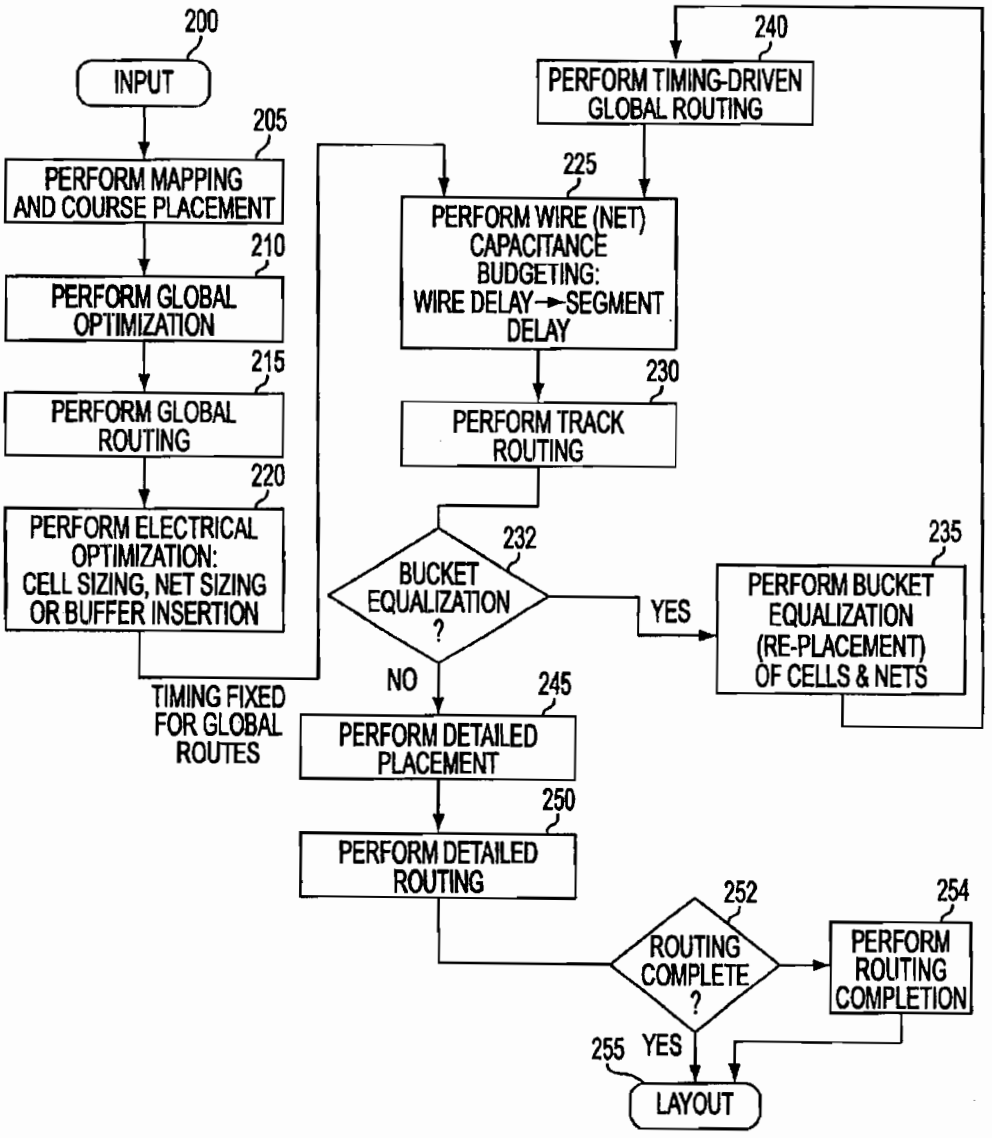

200
INPUT
205
PERFORM MAPPING AND COURSE PLACEMENT
210
PERFORM GLOBAL OPTIMIZATION
215
PERFORM GLOBAL ROUTING
220
PERFORM ELECTRICAL OPTIMIZATION: CELL SIZING, NET SIZING OR BUFFER INSERTION
TIMING FIXED FOR GLOBAL ROUTES
240
PERFORM TIMING-DRIVEN GLOBAL ROUTING
225
PERFORM WIRE (NET) CAPACITANCE BUDGETING: WIRE DELAY→SEGMENT DELAY
230
PERFORM TRACK ROUTING
232
BUCKET EQUALIZATION ?
YES
235
PERFORM BUCKET EQUALIZATION (RE-PLACEMENT) OF CELLS & NETS
NO
245
PERFORM DETAILED PLACEMENT
250
PERFORM DETAILED ROUTING
252
ROUTING COMPLETE ?
254
PERFORM ROUTING COMPLETION
255
YES
LAYOUT

FIG. 2

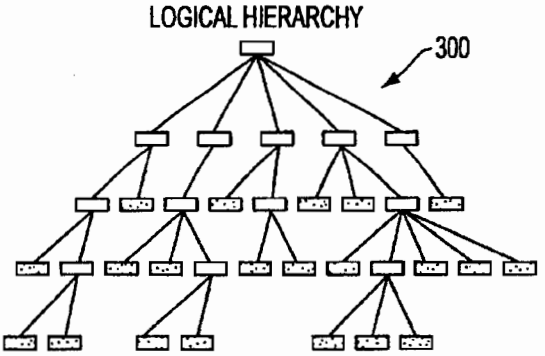


FIG. 3A

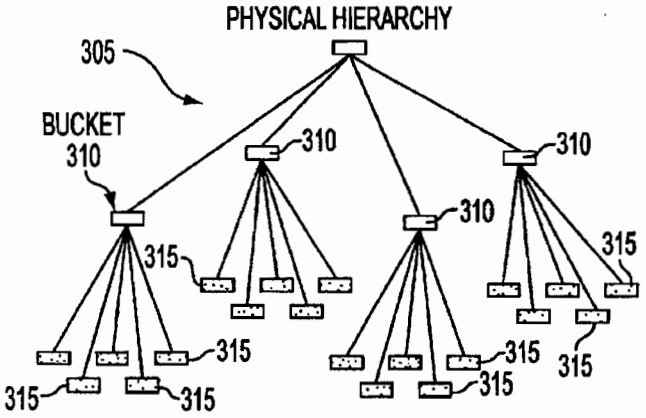


FIG. 3B

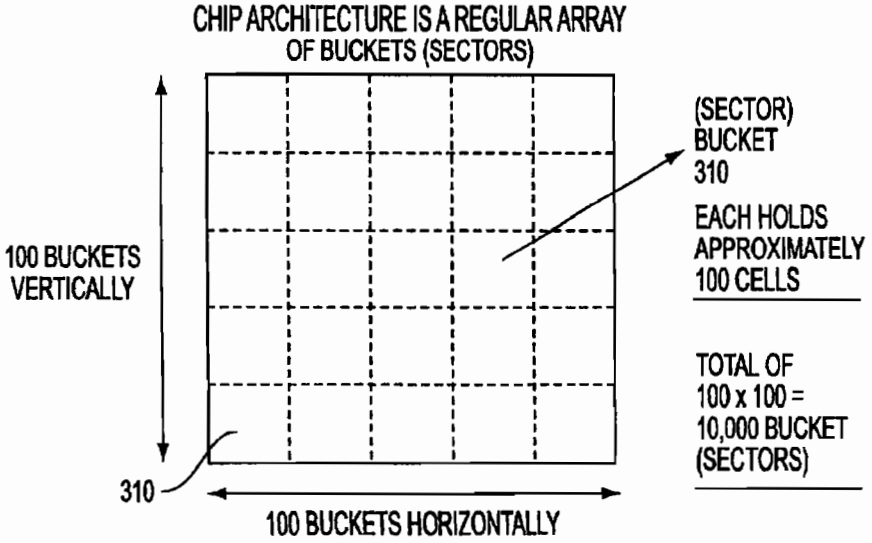
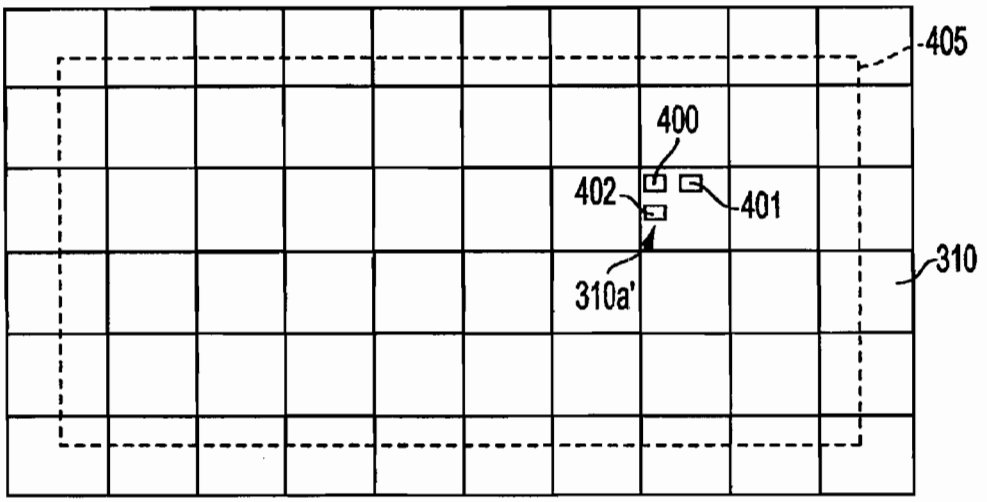


FIG. 3C



HIERARCHICAL MODEL ASSOCIATED IN A GIVEN BUCKET

FIG. 4

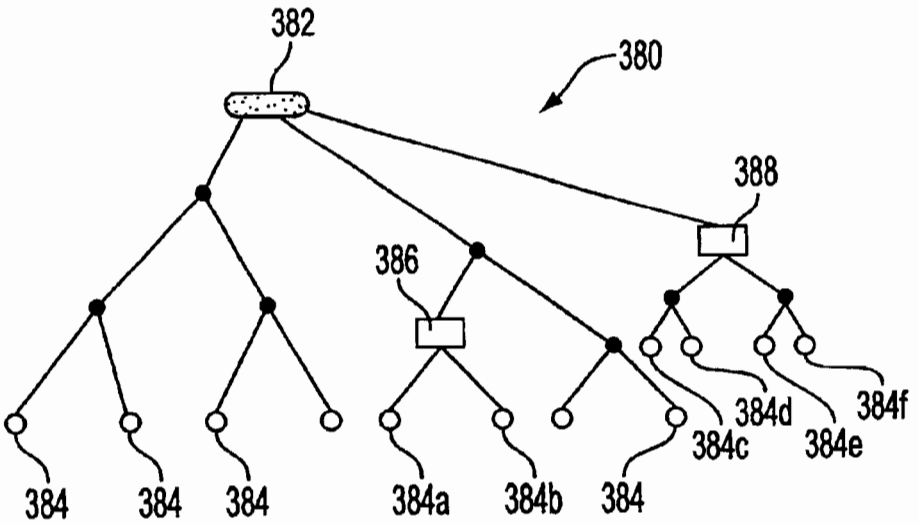

382
380
388
386
384
384
384
384a
384b
384
384c
384d
384e
384f

FIG. 5A

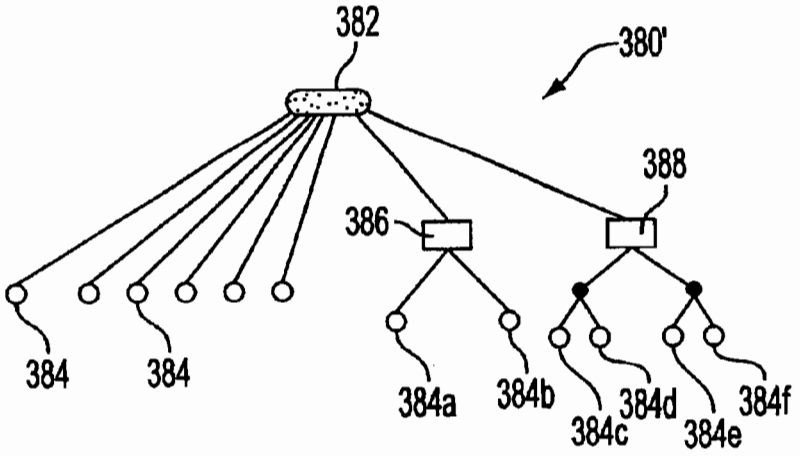

382
380'
388
386
384
384
384a
384b
384c
384d
384e
384f

FIG. 5B

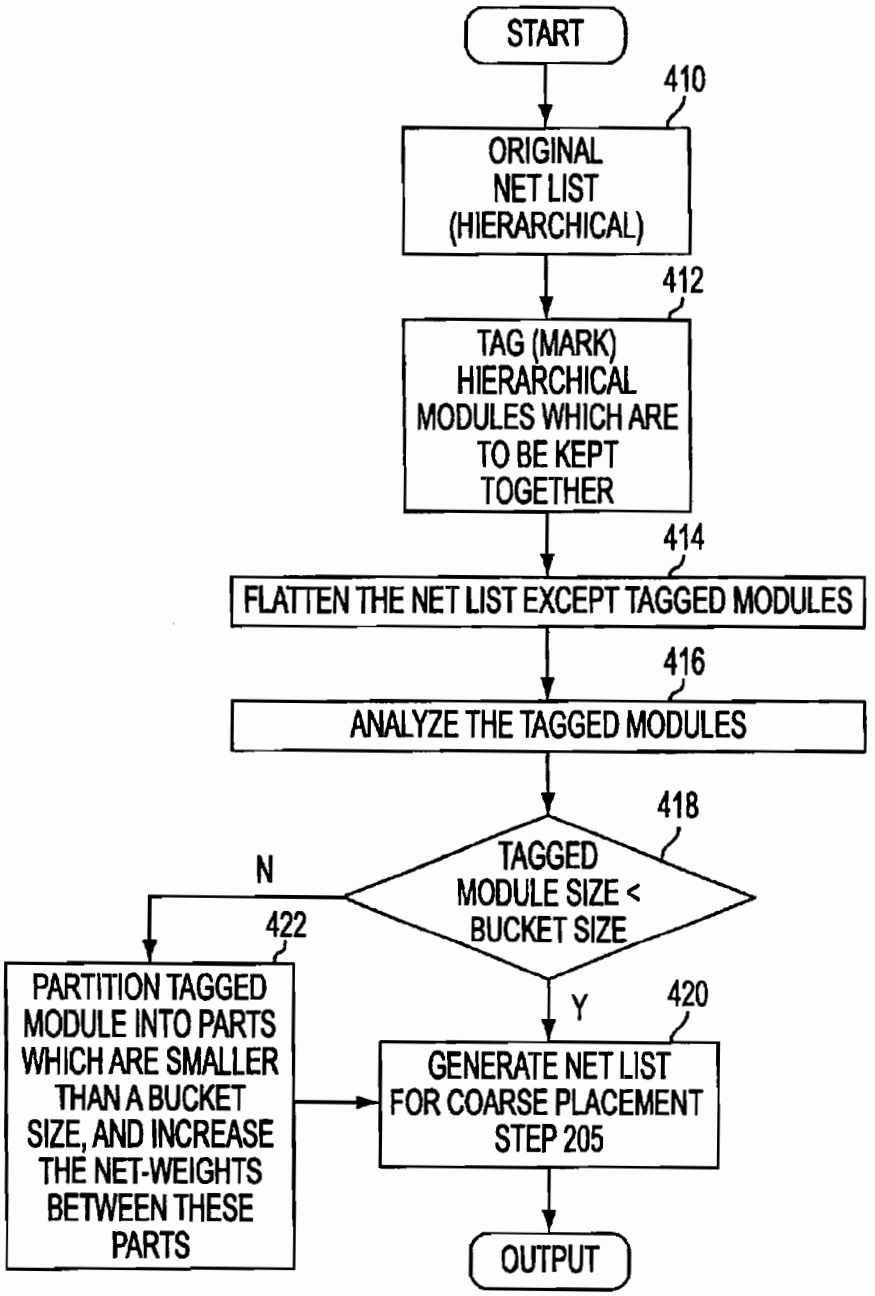

START
410
ORIGINAL NET LIST (HIERARCHICAL)
412
TAG (MARK) HIERARCHICAL MODULES WHICH ARE TO BE KEPT TOGETHER
414
FLATTEN THE NET LIST EXCEPT TAGGED MODULES
416
ANALYZE THE TAGGED MODULES
418
TAGGED MODULE SIZE < BUCKET SIZE
N
Y
422
PARTITION TAGGED MODULE INTO PARTS WHICH ARE SMALLER THAN A BUCKET SIZE, AND INCREASE THE NET-WEIGHTS BETWEEN THESE PARTS
420
GENERATE NET LIST FOR COARSE PLACEMENT STEP 205
OUTPUT

FIG. 5C

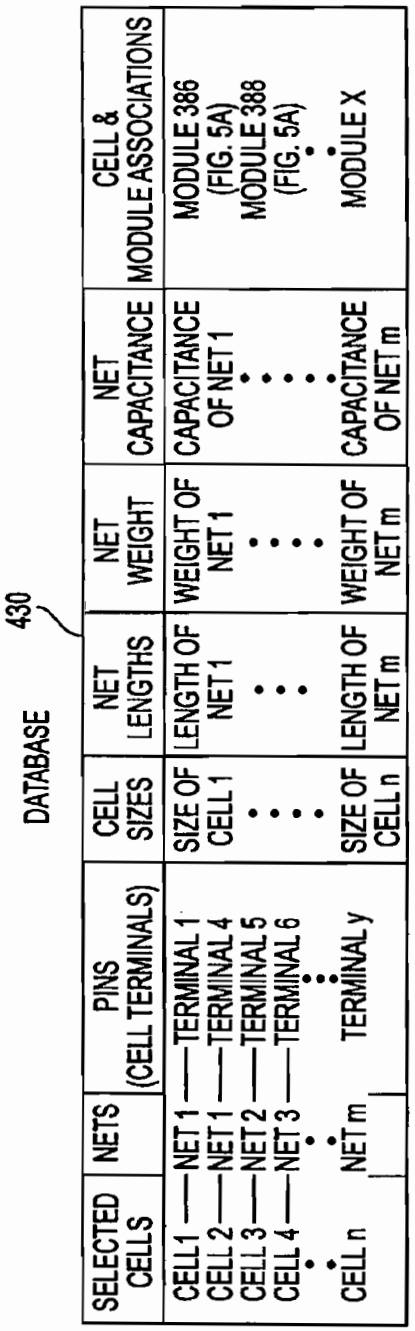

DATABASE
430
SELECTED CELLS
NETS
PINS (CELL TERMINALS)
CELL SIZES
NET LENGTHS
NET WEIGHT
NET CAPACITANCE
CELL & MODULE ASSOCIATIONS
CELL 1 — NET 1 — TERMINAL 1
CELL 2 — NET 1 — TERMINAL 4
CELL 3 — NET 2 — TERMINAL 5
CELL 4 — NET 3 — TERMINAL 6
CELL n NET m TERMINAL y
SIZE OF CELL 1
SIZE OF CELL n
LENGTH OF NET 1
LENGTH OF NET m
WEIGHT OF NET 1
WEIGHT OF NET m
CAPACITANCE OF NET 1
CAPACITANCE OF NET m
MODULE 386 (FIG. 5A)
MODULE 388 (FIG. 5A)
MODULE X

FIG. 5D

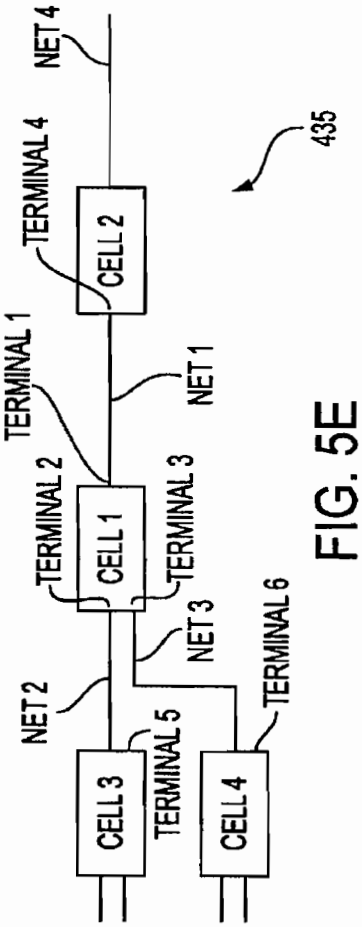

435
NET 4
TERMINAL 4
CELL 2
TERMINAL 1
NET 1
TERMINAL 2
CELL 1
TERMINAL 3
NET 2
NET 3
TERMINAL 5
TERMINAL 6
CELL 3
CELL 4

FIG. 5E

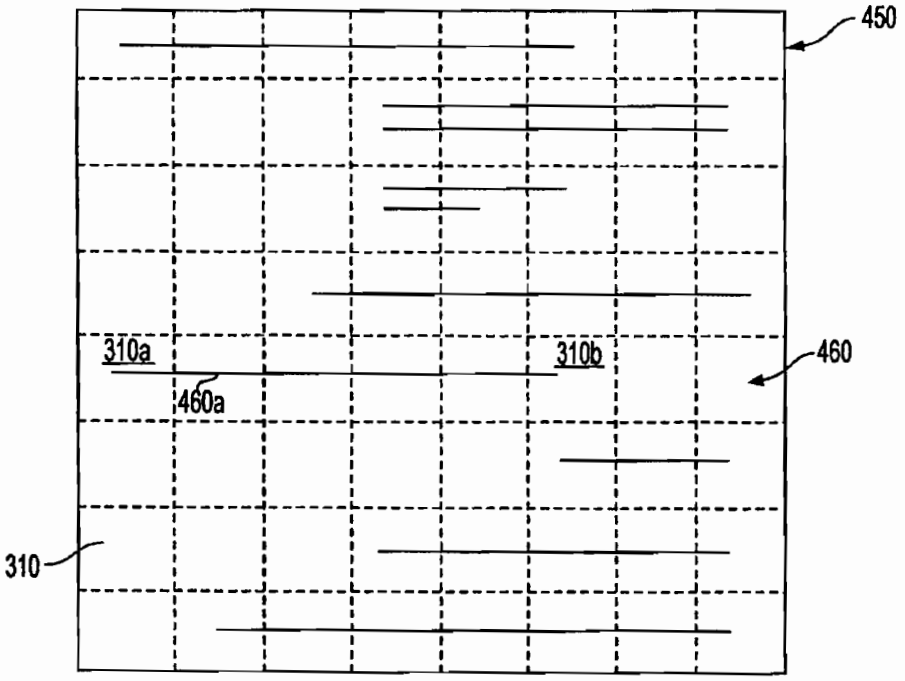


FIG. 6A METAL-1 LAYER 1

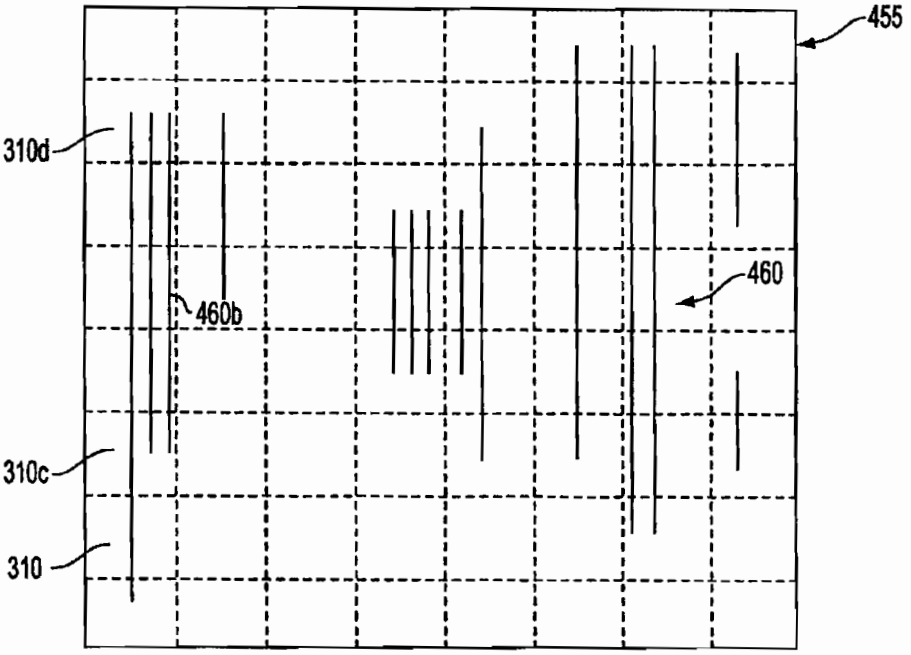


FIG. 6B METAL-2 LAYER 2

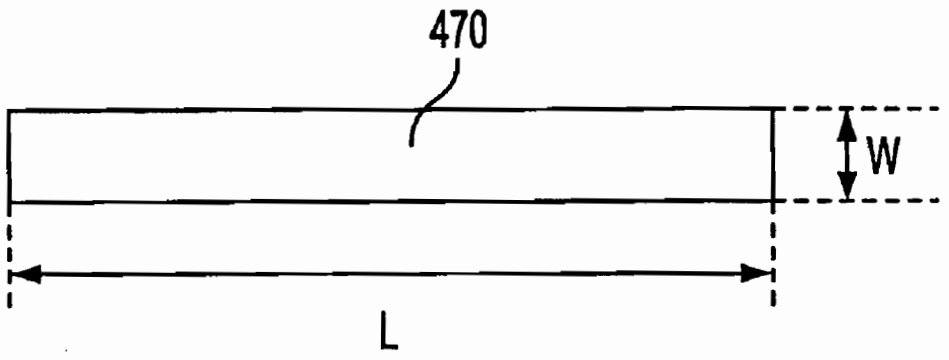


FIG. 7

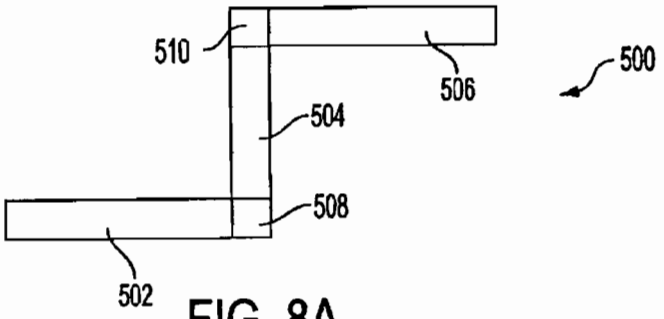


FIG. 8A

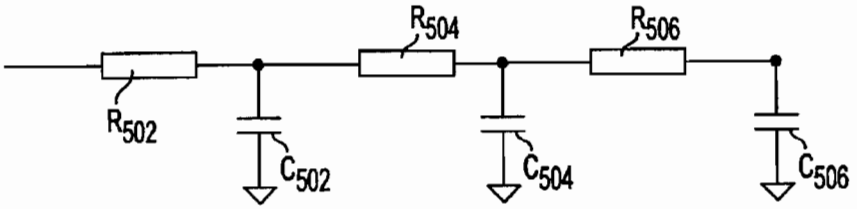


FIG. 8B

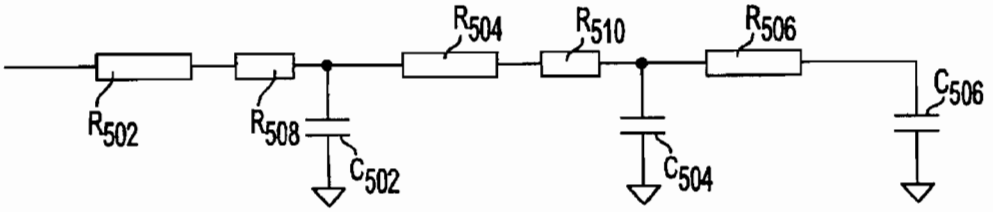


FIG. 8C

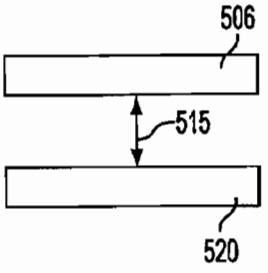


FIG. 8D

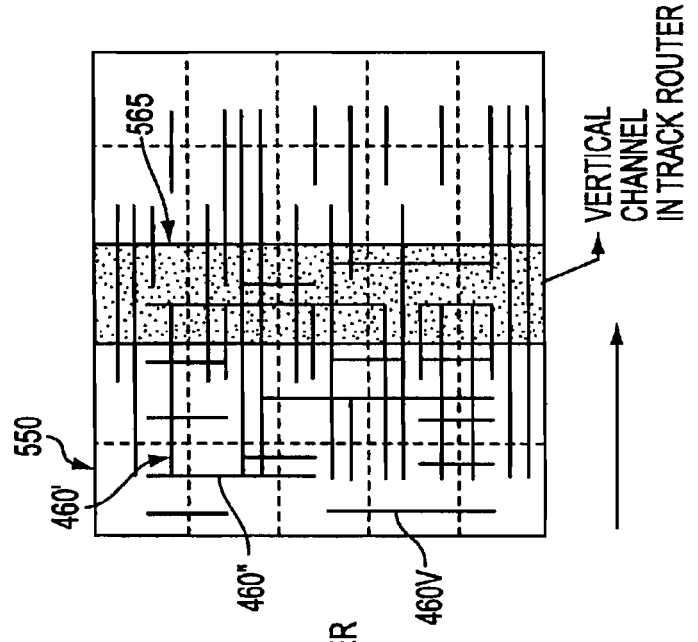

565
VERTICAL CHANNEL IN TRACK ROUTER
550
460'
460"
460V

FIG. 9B

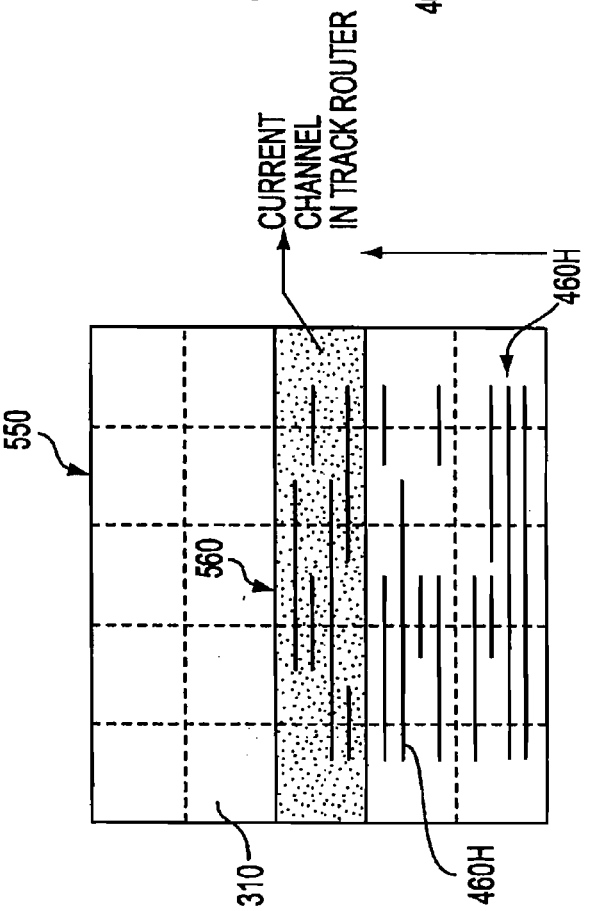

CURRENT CHANNEL IN TRACK ROUTER
460H
550
560
310
460H

FIG. 9A

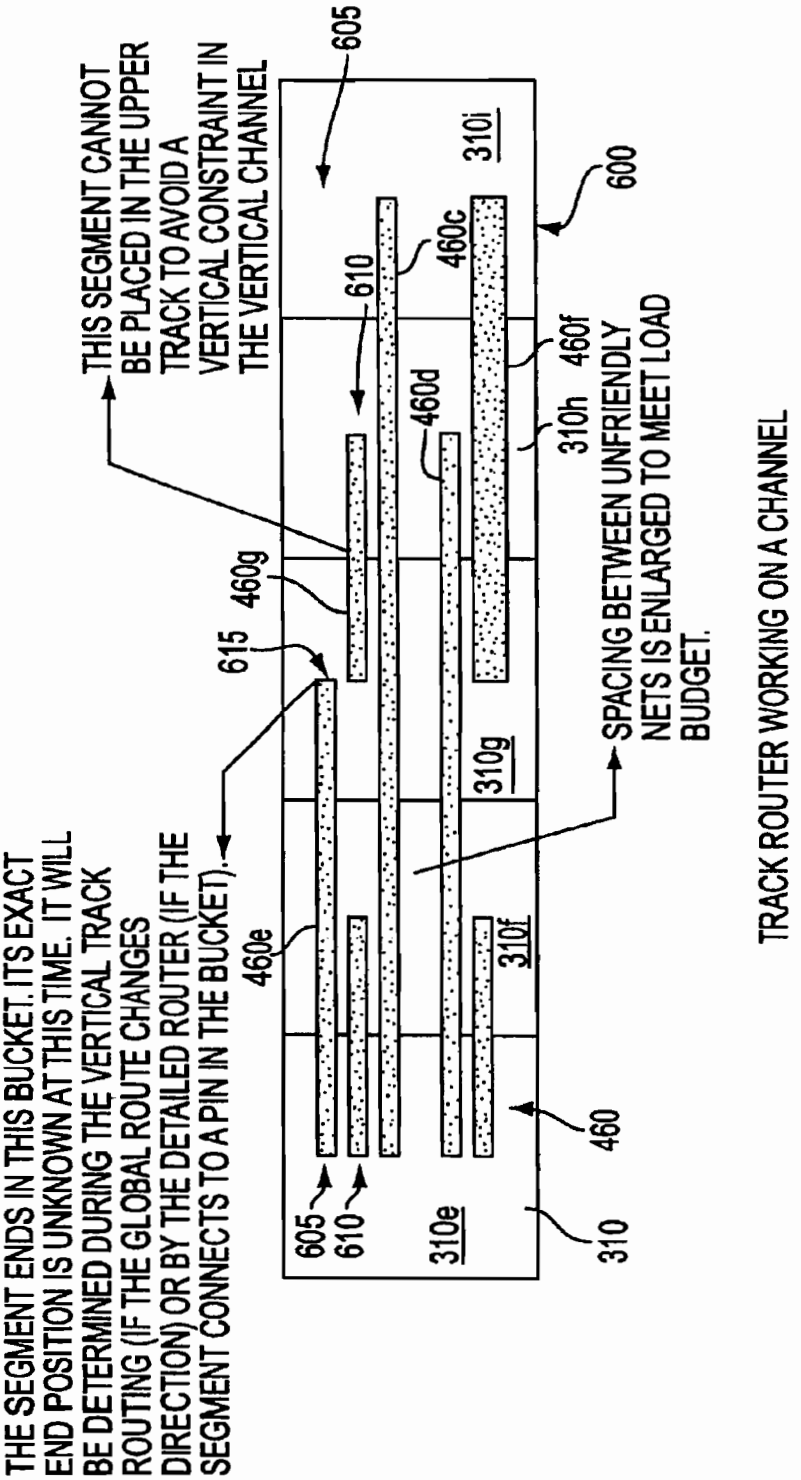

THE SEGMENT ENDS IN THIS BUCKET. ITS EXACT END POSITION IS UNKNOWN AT THIS TIME. IT WILL BE DETERMINED DURING THE VERTICAL TRACK ROUTING (IF THE GLOBAL ROUTE CHANGES DIRECTION) OR BY THE DETAILED ROUTER (IF THE SEGMENT CONNECTS TO A PIN IN THE BUCKET).
THIS SEGMENT CANNOT BE PLACED IN THE UPPER TRACK TO AVOID A VERTICAL CONSTRAINT IN THE VERTICAL CHANNEL
SPACING BETWEEN UNFRIENDLY NETS IS ENLARGED TO MEET LOAD BUDGET.
TRACK ROUTER WORKING ON A CHANNEL
605
610
615
460
460e
460g
460d
460c
460f
310h
310i
600
310
310e
310f
310g

FIG. 10

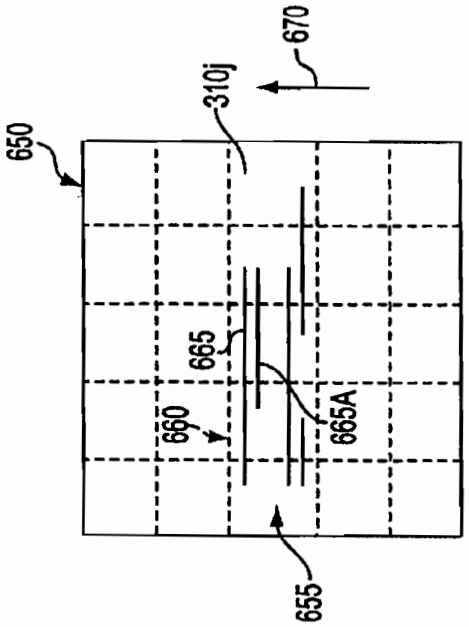


FIG. 11B

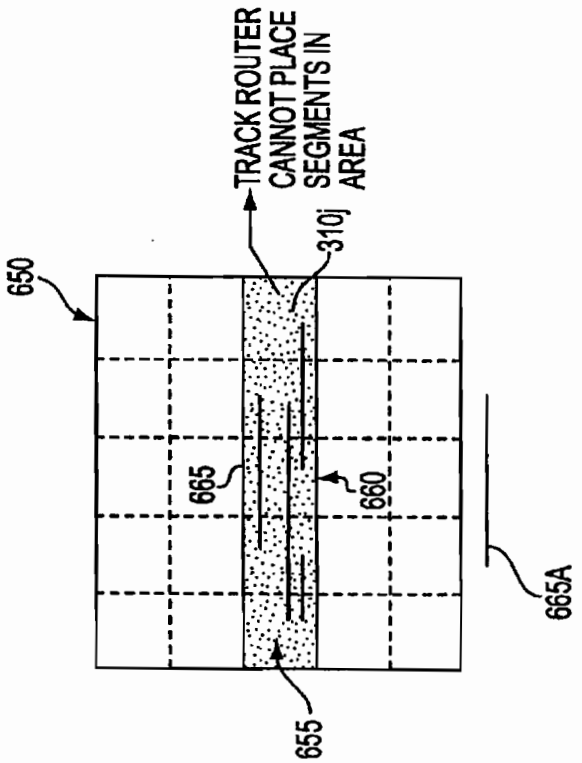
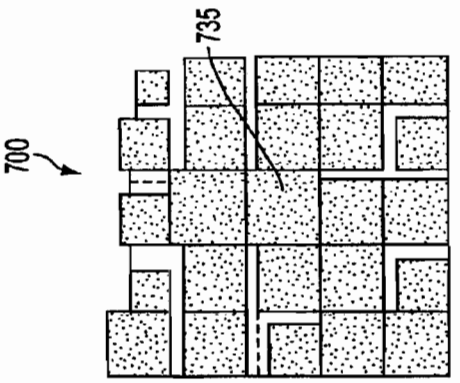


ILLUSTRATION: CREATING EXTRA AREA BY ENLARGING ALL BUCKETS IN A ROW.

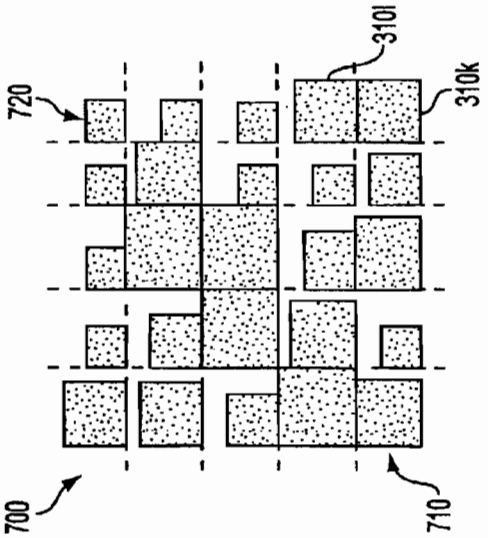
FIG. 11A



EQUALIZATION MOVES OVERFLOW CELLS TO NEIGHBORING CELLS IN AN ATTEMPT TO MINIMIZE THE CRITICAL PATH.

ILLUSTRATION OF THE EFFECT OF BUCKET EQUALIZATION

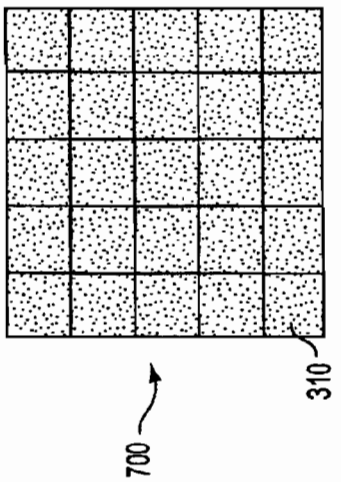
FIG. 12C



RE-OPTIMIZATION AND GLOBAL ROUTING CHANGES THE SIZE OF EACH OF THE BUCKETS SOMEWHAT, ENLARGING THE CIRCUIT.

ILLUSTRATION OF THE EFFECT OF BUCKET EQUALIZATION

FIG. 12B



INITIALLY THE BUCKETS ARE FILLED BY THE COARSE PLACER SUCH THAT THEIR SIZES ARE EQUAL.

ILLUSTRATION OF THE EFFECT OF BUCKET EQUALIZATION

FIG. 12A

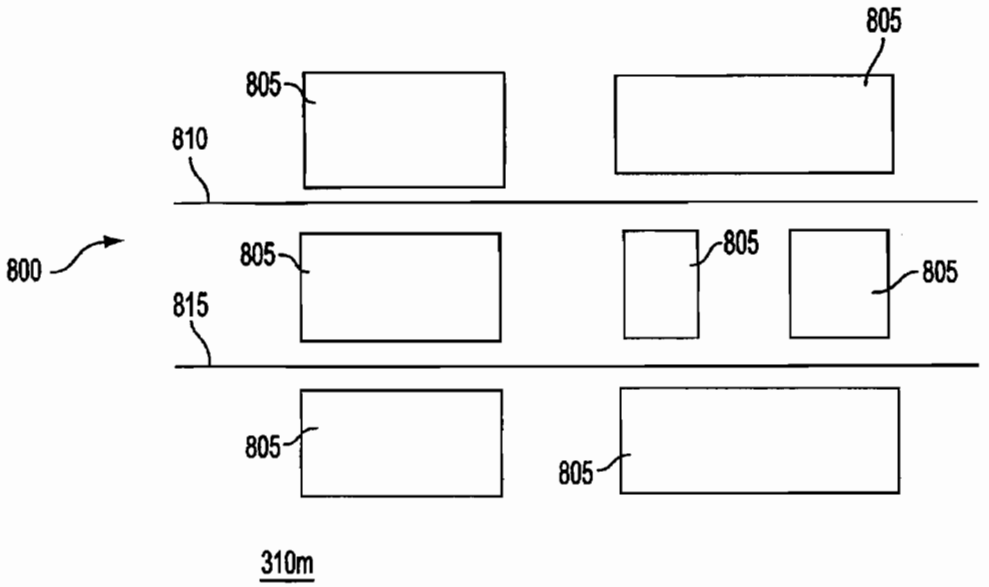
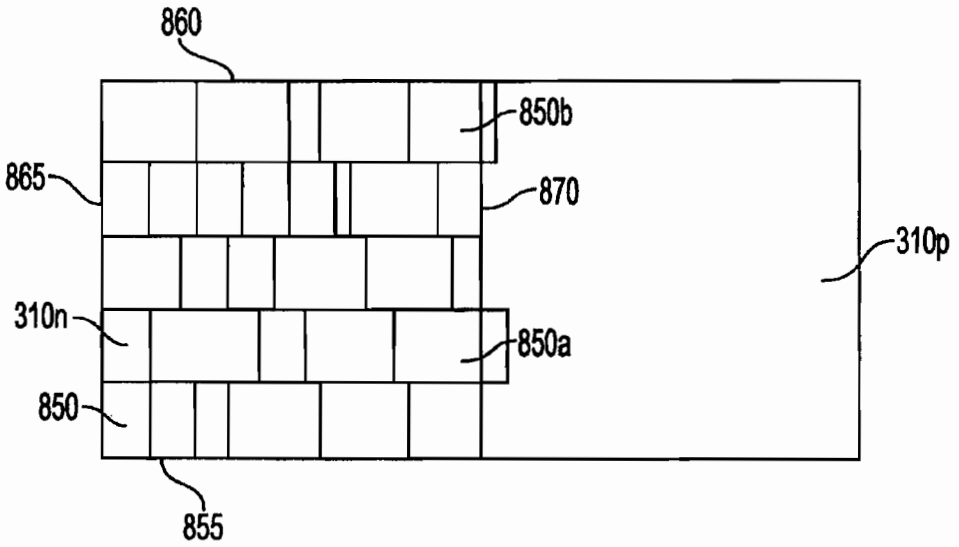


FIG. 13



A SINGLE BUCKET WITH STANDARD CELLS OVERLAPPING.

FIG. 14

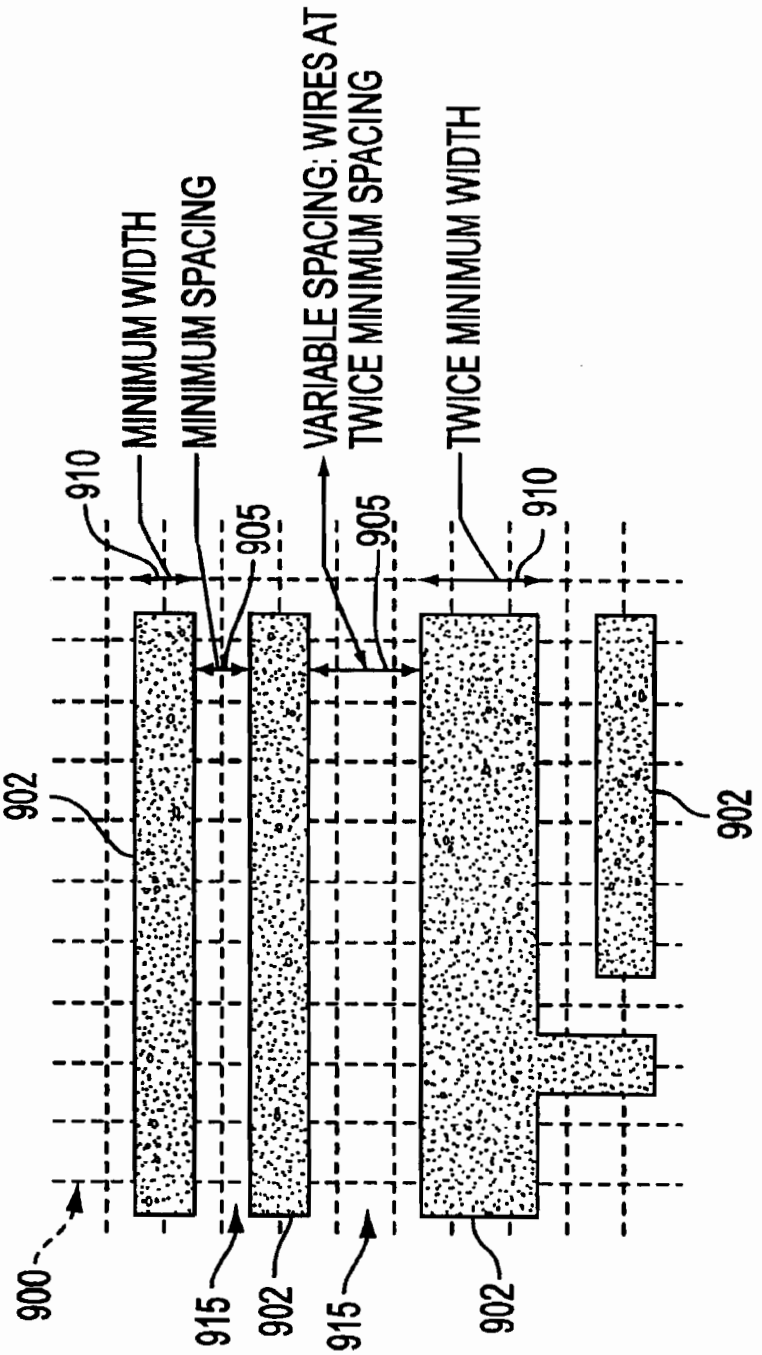

MINIMUM WIDTH
MINIMUM SPACING
VARIABLE SPACING: WIRES AT TWICE MINIMUM SPACING
TWICE MINIMUM WIDTH
910
905
905
910
902
902
900
915
902
915
902

FIG. 15

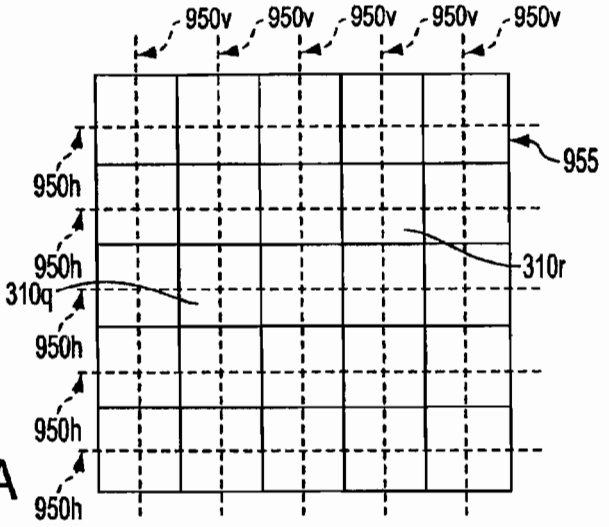


FIG. 16A

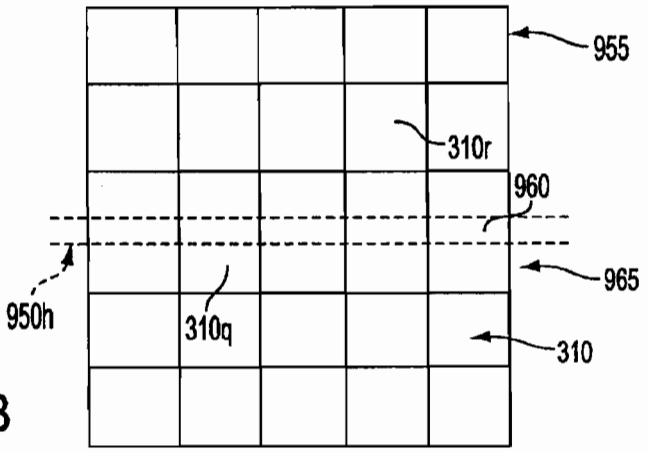


FIG. 16B

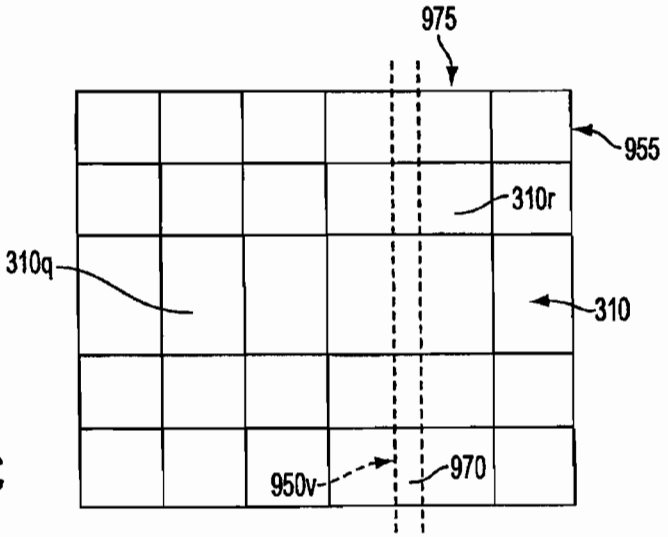
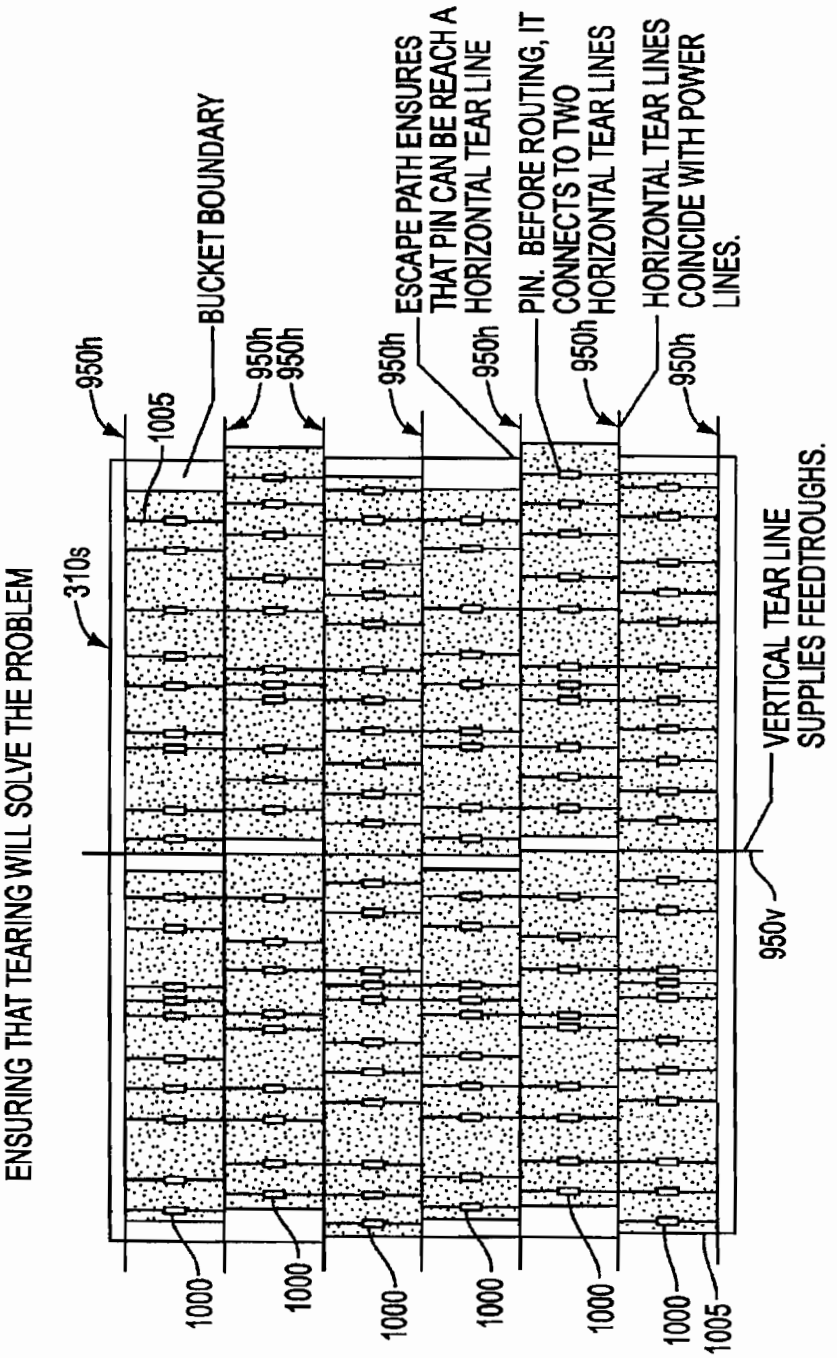


FIG. 16C



TEAR LINES IN A BUCKET

FIG. 17

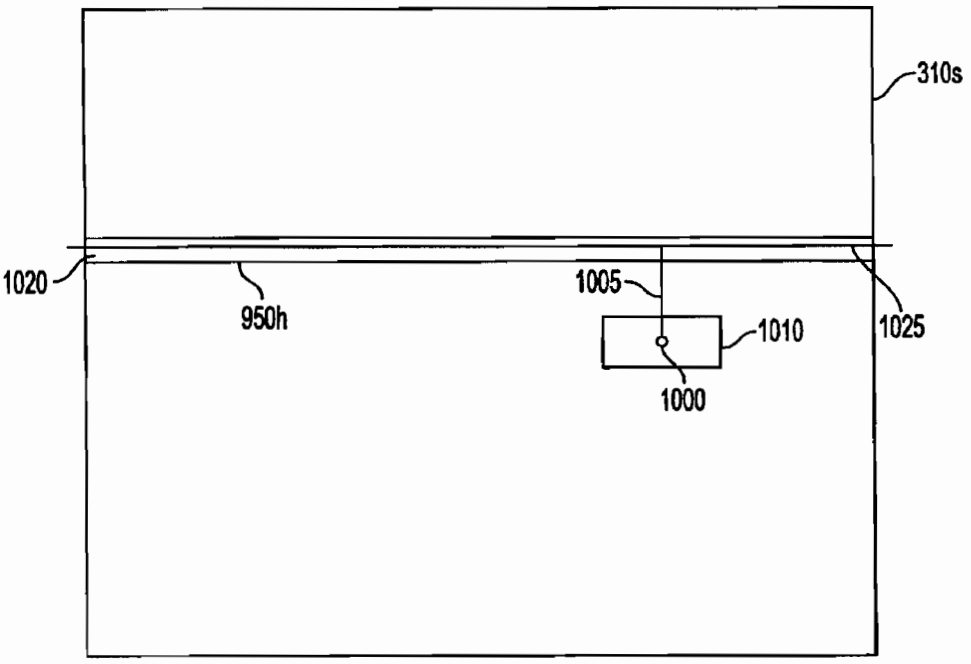


FIG. 18

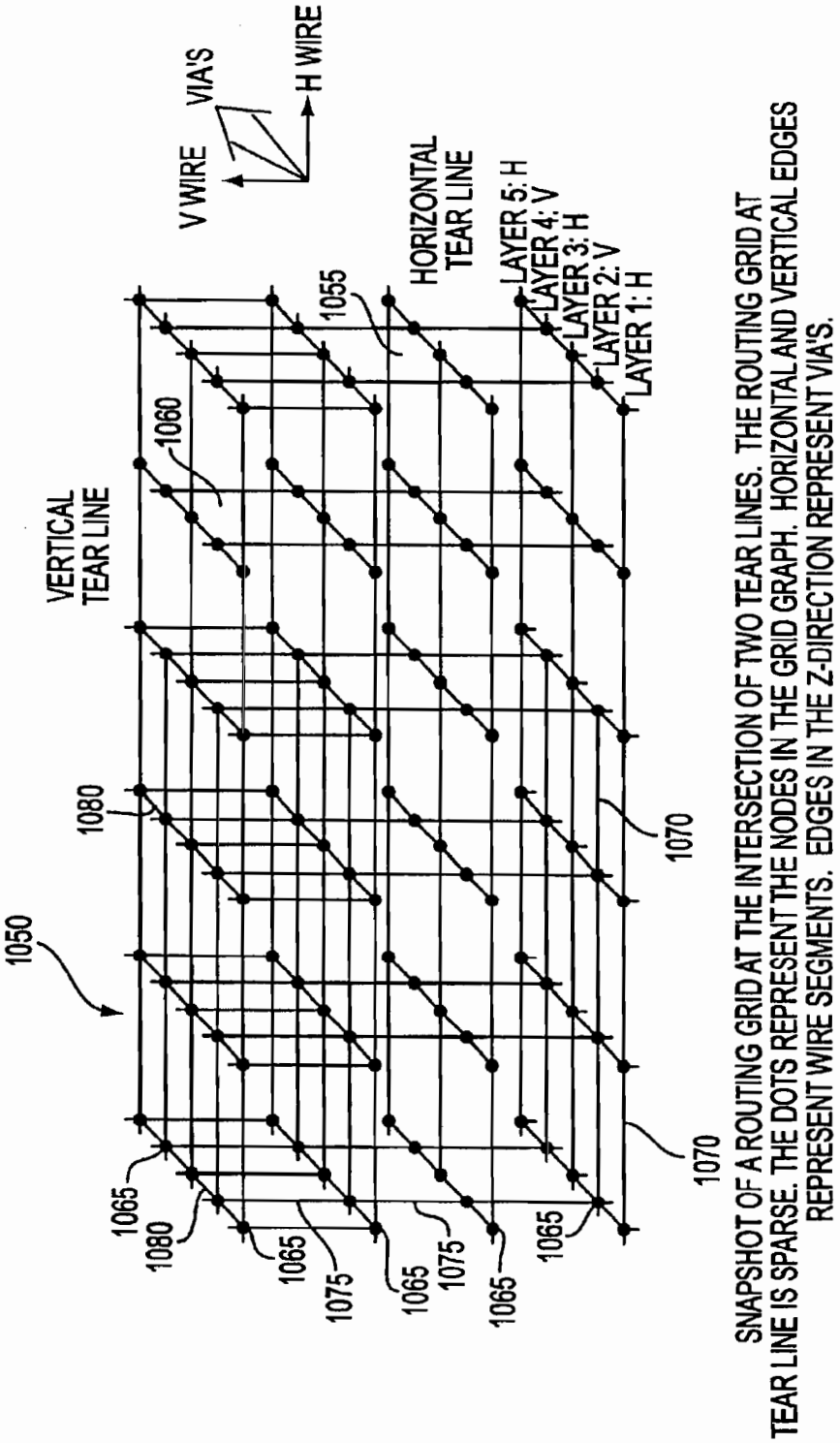


FIG. 19

SNAPSHOT OF A ROUTING GRID AT THE INTERSECTION OF TWO TEAR LINES. THE ROUTING GRID AT TEAR LINE IS SPARSE. THE DOTS REPRESENT THE NODES IN THE GRID GRAPH. HORIZONTAL AND VERTICAL EDGES REPRESENT WIRE SEGMENTS. EDGES IN THE Z-DIRECTION REPRESENT VIA'S.

METHOD OF DESIGNING A CONSTRAINT-DRIVEN INTEGRATED CIRCUIT LAYOUT

RELATED PATENT APPLICATION

This application claims the benefit of U.S. Provisional Application No. 60/068,827 filed Dec. 24, 1997. This patent application is commonly owned with the following co-pending related U.S. Patent Application entitled, "Timing Closure Methodology," filed concurrently herewith and fully incorporated herein by reference.

FIELD OF THE INVENTION

This invention relates generally to the field of integrated circuit design and more specifically to a method for routing the interconnections between components on an integrated circuit.

BACKGROUND OF THE INVENTION

A cell-based integrated circuit is formed by selecting a plurality of cells that represent components having different characteristics from one or more cell libraries, determining interconnects for the selected cells, and then placing and routing the interconnected selected cells to form the integrated circuit. For example, groups of cells may be interconnected to function as flip-flops, shift registers and the like. This process, overall, is conventionally described in terms of logic description, synthesis of that logic, and then placement and routing of the synthesized logic.

Electrical connections of individual components on integrated circuits are achieved using conducting paths (also called "wires" or "nets") between terminals of components which are to be connected. Automatic routing schemes are used to determine these conducting paths.

For more complex designs, there are typically at least four distinct layers of conducting medium available for routing, such as a polysilicon layer and three metal layers (metal-1, metal-2, and metal-3). The polysilicon layer, metal-1, metal-2, and metal-3 are all used for vertical and/or horizontal routing. It is a common practice to route each conducting path ("wire" or "net") by using one or more of the distinct layers of conducting medium, with one layer of a pair being reserved predominantly for connections running along the "x" direction and the other layer for connections running in the orthogonal or "y" direction. Some of the layers, such as the metal layers, are exclusively used for interconnection of components. The polysilicon layer may have a dual role, such as forming the gates of transistors as well as for interconnection of components.

An electrical connection between two nets on adjacent layers is implemented with a "via" which is an etched or drilled hole in the substrate for allowing a conductive path to extend from one layer to another layer.

Conventional design methodologies typically use a two-step process for determining the final size and location of each net. The first step is the global routing step for roughly determining wiring routes. The "rough" wiring pattern generated in this step is known as a "global route." Subsequently, a second detailed routing step for precisely determining a final routing pattern according to the global routes is used. This final routing pattern determined by the detailed routing step is known as a "detailed route."

In one conventional design methodology, die size is fixed and imaginary grid lines are used to partition the die into a matrix of blocks. Thus, the grid lines are used by the automated place and route equipment to assist in determining and then tracking of the location of the various nets and components that make up the integrated circuit.

With conventional "fixed die" design methodologies, complete routing of all nets cannot be ensured. Although complete routing can be ensured using channel routing techniques, channel routing techniques impose additional constraints and have their own problems.

Co-pending related U.S. Patent Application entitled "Timing Closure Methodology," which is fully incorporated herein by reference, describes a timing driven methodology, which methodology makes timing, but not area, a constraint. Accordingly, once placement of cells is determined based upon timing that has been fixed, the need exists to more efficiently route wires. Conventional routing techniques, including those mentioned above, can be used to route wires and connect cells whose placement has been determined by use of the timing driven methodology described above. However, conventional routing methodologies cannot efficiently ensure that the timing constraints are maintained. Accordingly, there is a need for a new routing method that works with the timing driven design methodology to provide more efficient and better results.

SUMMARY OF THE INVENTION

The invention broadly provides an automated method for designing an integrated circuit layout with a computer, based upon an electronic circuit description and upon a selected plurality of cells from a cell library, comprising the steps of:

(a) assigning each of the cells to one of a plurality of buckets designated on the integrated circuit layout, each of the cells being connected to one of the other cells;

(b) performing global routing to connect at least some of the selected cells of step (a) together such that global routes are formed to provide net topology information;

(c) performing track routing which sets the position of each of the global routes;

(d) performing detailed placement such that the positions of all selected cells are fixed within each of the buckets designated on the integrated circuit layout; and

(e) performing detailed routing such that detailed routes are formed to complete the integrated circuit layout.

The invention further provides an automated method for designing an integrated circuit layout with a computer, based upon an electronic circuit description and by using a cell library containing a selected plurality of cells, comprising the steps of:

(a) based upon a portion of a computer program that contains a sequence of instructions, assigning each of the cells to one of a plurality of buckets designated on the integrated circuit layout, each of the cells being connected to one of the other cells based upon the electronic circuit description input to the computer;

(b) performing global routing to connect at least some of the selected cells of step (a) together such that global routes are formed to provide net topology information;

(c) performing track routing which sets the position of each of the global routes;

(d) performing detailed placement such that the positions of all selected cells are fixed within each of the buckets designated on the integrated circuit layout; and

(e) performing detailed routing such that detailed routes are formed to complete the integrated circuit layout.

A automated method for designing an integrated circuit layout by using a computer and based upon an electronic circuit description, comprising the steps of:

(a) using a portion of a computer program that contains a sequence of instructions, placing a first plurality of cells in designated initial positions;

(b) forming a plurality of global routes based upon the designated initial positions of the first plurality of cells of step (a);

(c) fixing the positions of the global routes of step (b);

(d) placing a second plurality of cells based upon the positions of the global routes of step (c); and

(e) placing a plurality of detailed routes to complete the integrated circuit layout.

The present invention makes possible an advantage of providing a methodology for setting and achieving timing constraints for an-integrated circuit.

The invention also makes possible an advantage of providing a method to efficiently place cells and route wires based on fixed timing constraints for the desired circuit.

The invention further places timing driven routing at selected stages in the design process so as to optimize routing and placement results. The placement and routing results are further optimized since detailed placement is adapted based on the fixed positions of the.global routes.

The invention makes possible another advantage of enabling a faster process at later stages of the design process.

The invention makes possible an additional advantage of permitting capacitance control on a net-per-net basis which can lead to reduced die sizes.

Still another advantage made possible by the invention is the achievement of routing completion for an integrated circuit layout under constraints (e.g., timing constraints).

The invention also make possible an advantage of enabling detailed placement to be re-performed so as to achieve routing completion for the desired circuit.

The list of possible advantages and benefits above is not necessarily exhaustive.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a schematic diagram of a host computer system which is capable of implementing the present invention;

FIG. **2** is a flowchart describing a method of the present invention;

FIG. **3A** is a schematic block diagram of a logical hierarchy of a circuit in accordance with the invention;

FIG. **3B** is a schematic block diagram of a physical hierarchy of a circuit in accordance with the invention wherein the physical hierarchy includes "buckets" with assigned cells;

FIG. **3C** is a partial top elevational view of a plurality of "buckets" wherein cells are placed or assigned inside the buckets;

FIG. **4** is a schematic diagram showing a plurality of hierarchical modules (in a circuit) as being grouped together in a given bucket;

FIG. **5A** is a hierarchical netlist comprising a top hierarchical level, intermediate levels, and a low hierarchical level (which comprises cells);

FIG. **5B** shows a resulting netlist if the hierarchical netlist of FIG. **5A** is flattened;

FIG. **5C** is a flowchart describing a method for generating a netlist for use in a coarse placement step while grouping together pre-determined hierarchical modules;

FIG. **5D** is a database representation which contains parameters and component information which are used with the computer system of FIG. **1** for designing an integrated circuit layout in accordance with the present invention;

FIG. **5E** is partial schematic diagram of a circuit which can be designed based on the parameters and component information stored and manipulated in the database representation of FIG. **5D**;

FIG. **6A** is a partial top elevational view of a first interconnection layer image to be formed in the semiconductor die wherein the image includes horizontal global routes;

FIG. **6B** is a partial top elevational view of a second interconnection layer image to be formed in the semiconductor die wherein the image includes vertical global routes;

FIG. **7** is a partial top elevational view of a global route having a length L and a width W;

FIG. **8A** is a partial top elevational view of a net which results from the global routing step and which comprises three segments;

FIG. **8B** is a schematic diagram comprising discrete resistive and capacitive elements which represent an Elmore delay model of the net of FIG. **8A**;

FIG. **8C** is a schematic diagram which represent a variation of the Elmore delay model of FIG. **8B**;

FIG. **8D** is a partial top elevational view of two adjacent net segments separated by a given distance which determine net-to-net capacitance;

FIG. **9A** is a partial top elevational view of an interconnection layer image as shown during a horizontal track routing step;

FIG. **9B** is a partial top elevational view of an interconnection layer image as shown during a vertical track routing step;

FIG. **10** is a top elevational view of track routing on a channel whereby vertical constraints have been eliminated so that empty spaces are present in some of the buckets;

FIG. **11A** is a partial top elevational view of an interconnection layer image wherein the track router is unable to place all of the net segments in a given channel;

FIG. **11B** is a partial top elevational view of an interconnection layer image wherein the channel is enlarged to create additional space to fit net segments;

FIG. **12A** is a partial top elevational view of an interconnection layer image having equal sized buckets;

FIG. **12B** is a partial top elevational view of the interconnection layer image of FIG. **12A** after optimization and track routing change the size of each of the buckets, thereby enlarging the size of the desired circuit;

FIG. **12C** is a partial top elevational view of the interconnection layer image of FIG. **12B** after bucket equalization is performed, whereby overflow cells are moved adjacent to neighboring cells in an attempt to minimize critical areas (which impact the size of the circuit);

FIG. **13** is a top elevational view of a A placement in a portion of a bucket wherein the cells are placed close to the global routes and are oriented appropriately in the bucket portion;

FIG. **14** is a top elevational view of buckets wherein some standard cells in one bucket overlap into the other bucket since some of the standard cells have variable widths;

FIG. **15** is a top elevational view of a fine grid which serves to guide detailed routing in each of the buckets;

FIG. **16A** is a top elevational view of a bucket array and tear lines which can be formed in the bucket array to create additional space for routing;

FIG. **16B** is a top elevational view of the bucket array of FIG. **16A** whereby a tear line is created along a row of

buckets to form additional routing space for achieving routing completion for a portion of the circuit;

FIG. **16C** is a top elevational view of the bucket array of FIG. **16B** whereby a second tear line is created along a column of buckets to form additional routing space for achieving routing completion for another portion of the circuit;

FIG. **17** is a top elevational view of a given bucket showing a plurality of tear lines and escape paths which can be formed in a bucket to achieve routing completion;

FIG. **18** is a top elevational view of a given bucket showing the achievement of routing completion for the circuit by coupling a cell terminal to a net which is formed in the tear line; and

FIG. **19** is a perspective view of a portion of the routing grid at the intersection of two tear lines wherein the routing grid is sparse at the tear lines.

DETAILED DESCRIPTION OF THE INVENTION

Those of ordinary skill in the art will realize that the following description of the present invention is illustrative only and not in any way limiting. Other embodiments of the invention will readily suggest themselves to those skilled in the art.

Referring in detail now to the drawings wherein similar parts or steps of the present invention are identified by like reference numerals, there is seen in FIG. **1** a schematic diagram of a host computer system **150** which is capable of implementing the present invention. The host computer system **150** preferably includes a computer **155**, a display **160**, a printer **165** and a plotter **170**. The computer **155** is typically a personal computer or workstation. The display **160** displays both graphical and textual materials relating to the design of integrated circuits. Also included in the computer system **150** is a keyboard **180** and a pointing device **185** such as a "mouse". When operating with software tools used in integrated circuit design, the computer system **150** essentially becomes a series of electronic circuits for accomplishing specific design functions. One example of such software tools is Aquarius, which is commercially available from Avant! Corporation of Fremont, Calif.

FIG. **2** is a flowchart describing an overview of the automated method of the present invention that is implemented using a computer program containing sequences of instructions that implement each of the various steps described hereinafter. The automated program thus implements a method that includes the following steps below. It is noted further that in the flowchart of FIG. **2**, each of the steps has its own distinct computer instructions (i.e., each of the steps **200–254** comprises further steps). Initially in step **200**, a netlist or other electronic circuit description is received as an input in the placement and routing process in accordance with the invention. This netlist includes information on all the interconnections between the terminals of the cells in a specified circuit to be implemented. Based on the provided netlist, in step **205** the specified circuit is mapped to a library of cells (not shown). Coarse placement is also performed in step **205** whereby each cell (which has a specified function) is assigned to or associated with a specific bucket (e.g., bucket **310** in FIGS. **3A–3C**) which represents bucket on a semiconductor die.

In step **210**, a global optimization is performed whereby the specified circuit is partially remapped. This optimization is based on the initial positioning of cells (as performed in the coarse placement step **205**) and may change the logic structure of the specified circuit. In step **215**, global routing is performed whereby the net topologies (i.e., net lengths) and net interconnection layer (i.e., conducting medium) assignments for cells that are located in different buckets are determined. The net topology information which results from the global routing step **215** is used as a basis when performing optimizations in the electrical optimization step **220**. Predetermined timing constraints (which have been predetermined prior to coarse placement **205**) are satisfied in the electrical optimization step **220**.

In step **225**, net capacitance budgeting is performed whereby the capacitance of each net may be adjusted in order to meet the capacitance constraints of the nets. In the track routing step **230**, the routing order is determined for the global routes (wherein the topologies of the global routes have been determined in the above global routing step **215**). Additionally, in the track routing step **230**, modified positions are determined for the global routes which will be placed on the semiconductor die. The modified positions of the global routes take into account the cross-talk capacitance of each net such that timing constraints for the designed circuit are met.

In step **232**, it is determined if bucket equalization is to be performed, and this step involves examining the differences in bucket sizes. As best shown in FIGS. **12A–12C**, bucket equalization is performed to reduce the size of the largest bucket in a given row (e.g., row **710**) or a given column so that the chip size is optimized. If bucket equalization is performed, as shown by step **235**, then certain cells may be moved to neighboring buckets (see FIGS. **12A** to **12C**). Since the bucket equalization step **235** changes the positions of some of the cells, it is necessary to again perform a global routing step. This is shown in FIG. **2** as a timing-driven global routing step **240**. Additionally, the capacitance budgeting step **225** and track routing step **230** are again performed.

In step **245**, detailed placement is performed whereby the final position is determined for each of the cells within each bucket to be placed on the semiconductor die. It is noted that cells that are connected to other cells entirely within a bucket have not yet been operated upon to ensure that timing constraints are met for such cells. In step **250**, detailed routing is performed, while in step **252** it is determined if routing completion is achieved. If not, then the routing completion step **254** is performed. To insure routing completion, the chip layout may be torn at "tear lines" (see FIGS. **16A–16C**) to create additional space for routing. The layout is then generated in step **255**.

As described above and shown in FIG. **2**, the placement of cells is divided into two separate stages: coarse placement step **205** and detailed placement step **245**. Additionally, the determination of global routes is divided into two stages: the global routing step **215** and track routing step **230**. Additionally, according to the present invention, the global routing step **215** is performed before cell placement positions have been fixed or finalized. The advantages of separating cell placement into two stages and global routing into two stages are elaborated upon more fully hereinafter. In contrast, in conventional place and route methodologies, cell placement is performed in one step and global routing is performed in another step. For example, in conventional methodologies, cell placement is performed, followed by an optimization step and global routing.

Coarse Placement

The coarse place step **205** is performed based on the netlist or electronic circuit description input of step **200**. As known to those skilled in the art, the netlist is a set of

information, typically a "recipe" in the form of a data structure stored in a computer. The netlist denotes all the interconnections between the terminals of the circuit components in a specified circuit. Other information which serves as input in step **200** may include the position in which each cell should be disposed on the semiconductor die, the shape of each cell, positions where nets are forbidden to pass through in the semiconductor die, and/or process design rules which specify specific technology information such as minimum widths and spacing.

In the coarse placement step **205**, an approximate (initial) placement of cells **315** (FIG. 3B) is determined so as to obtain an initial or preliminary measurement of the nets for the desired circuit, while at the same time leaving sufficient flexibility in the netlist to permit electrical optimizations (e.g., step **220** in FIG. **2**) and other synthesis steps. Preferably, the coarse placer determines the intended placement of cells **315** (FIG. 3B) such that the total net length is minimized while the wiring density is spread evenly over the semiconductor die. The coarse placer need not be timing-driven, since timing violations which occur can be compensated in the electrical optimization step **220**. The initial placement of cells **315** (as determined by the coarse placer) and the global routing topologies (as determined from the global routing step **215**) serve as inputs for the electrical optimization step **220**.

FIGS. 3A to 3C show conceptually how a circuit design is transformed from a logical hierarchy **300** to a physical hierarchy **305** during the coarse placement step **205**. In the physical hierarchy **305**, the intermediate logic levels in the logical hierarchy **300** are associated with or grouped in buckets (sectors) **310** wherein each bucket **310** holds, for example, about one-hundred (100) cells **315**. The cells **315** will form the designed circuit. The buckets **310** represent sectors in the placement area of the semiconductor die. The number of cells **315** which can be placed (associated) within a given bucket **310** can range, preferably, from about 20 cells to about 200 cells. Each cell **315** is assigned to a particular position in a given bucket **310**. Furthermore, the cell sizes have not yet been finalized at this stage of the design process, since cell sizes may be adjusted in the electrical optimization step **220** (FIG. **2**).

To perform the coarse placement step **205**, a conventional placer may be used. General placement techniques known to those skilled in the art are used for placing the cells **315** in given buckets **310**. Such general placement techniques include min-cut, simulated annealing, and quadratic placement.

The buckets **310** are arranged, for example, in an array as shown in FIG. 3C. If the designer chooses to keep a given group of cells **315** together, then the given group of cells **315** are grouped in the same bucket **310** or in neighboring buckets. Preferably, each bucket **310** is sized small enough such that cell placement anywhere within it has an insignificant effect on timing. However, the size of a given bucket **310** should also be large enough to accommodate the re-mapping and re-sizing of cells **315** contained in the given bucket. A given bucket must also be large enough to enable a meaningful detailed placement of the cells **315** inside a given bucket **310**. The dimensions of a given bucket **310** and a given bucket array is programmed in the placer which performs the coarse placement step **205**.

Pre-routes and pre-places are also read into the bucket structure in order to drive the placer and global router. According to the invention, pre-placement of standard cells and pre-routing of specific nets are permitted. The cells can be shifted apart slightly in positions and some nets can be stretched slightly. These pre-placements and pre-routing are formed in the bucket **310** structure and thus reduce the capacity of certain buckets. The pre-routed nets can include the power nets and the clock nets.

The coarse placement step **205** provides an advantage of permitting the placer to place cells at a faster rate, as compared to conventional approaches, as best illustrated by the following comparison. In a one-hundred-by-one-hundred (100x100) bucket array, the placer can place a cell after consideration of 10,000 (i.e., 100x100) different possibilities. In contrast, in a conventional design approach, a placer can place a cell after consideration of, for example, one-million different positions if the designed circuit will utilize one-million cells.

Referring now to FIG. **4**, certain hierarchical modules (models) **400**, **401** and **402** may be grouped together by forcing them to be placed in the same bucket **310***a*'. FIGS. **5A–5C** show a method for performing clustering in accordance with the present invention which can assist in keeping cells that are desirably kept together in the same bucket. This clustering step can be performed prior to the coarse placement step **205** (FIG. **2**). Referring first to FIG. **5A**, there is seen a hierarchical netlist **380** comprising a top hierarchical level **382** which represents the entire integrated circuit, and a low hierarchical level which comprises cells **384** which form the integrated circuit. The hierarchical netlist **380** further comprises intermediate hierarchical levels such as the modules **386** and **388** which contain cells that are desirably kept together physically. In FIG. **5A**, the cells **384***a* and **384***b* are grouped (associated) in module **386**, while the cells **384***c*–**384***f* are grouped (associated) in module **388**. If the hierarchical netlist **380** is flattened, then it is transformed, for example, into the netlist **380'** as shown by FIG. **5B**.

Referring now to FIG. **5C**, there is seen a method for generating a net list for use in a coarse placement while keeping together desired hierarchical modules. In step **410** the original hierarchical netlist **380** is provided. In step **412**, the circuit designer can conventionally tag (i.e., mark) given modules (e.g., modules **386** and **388**) in the intermediate hierarchical levels, if the circuit designer intends to keep together components which form the given modules. In step **414**, the hierarchy of the netlist **380** is flattened except for the tagged modules of step **412**. The flattened netlist **380'** is shown in FIG. **5B**. In step **416**, the tagged modules **386** and **388** are analyzed. If in step **418** each of the tagged modules **386** and **388** fits within a bucket size, then a net list is generated in step **420** for coarse placement. If in step **418** a tagged module does not fit within a bucket size, then the tagged module is partitioned into parts which are smaller than a bucket size (step **422**). The net weights are increased between the partitioned parts so that the parts are placed closed to each other in the layout.

A "bottom-up clustering step" can be performed before the coarse placement step **205** (FIG. **2**) so as to speed up the placement. This bottom-up clustering step will "greedily" force logic together whereby predetermined portions (or highly connected portions) of the desired circuit **405** (FIG. **4**) are associated together with a higher priority as compared to other portions of the desired circuit. Typically, logic which is forced together ranges from about 10 to about 25 cells and occupies about ¼ of the space in a given bucket.

For the case when the size of a hierarchical module (in a desired circuit shown by dashed box **405** in FIG. **4**) is larger in size than a given bucket **310**, the hierarchical module can have different parts thereof associated with a plurality of given buckets **310**. The plurality of given buckets **310**

containing the hierarchical module can then be kept together by specifying a stronger connection constraint between the given plurality of buckets **310**. A stronger connection constraint between a given plurality of buckets **310** (or between certain cells in different buckets) is accomplished as follows. Coarse placers have an objective of assigning initial cell positions so that the net lengths are minimized in the desired circuit. By specifying to the coarse placer that a given net coupled between first and second cells should be shorter than another net coupled between first and third cells, the coarse placer will group the first and second cells closer together than the first and third cells, regardless of whether the first and second cells are in the same bucket. The specified length of a given net is preferably given relatively, by using a parameter that indicates, relative to the other nets how long it should be, i.e., a connection constraint. Thus, if the weight of the given net is increased, then the specified permissible length of the given net is increased.

Global Optimization

The global optimization step **210** involves a timing-driven, partial re-mapping of the desired circuit and is based on the cell placement information obtained as a result of the coarse placement step **205**. (The timing requirements or constraints have been determined before the coarse placement step **205** of FIG. **2**). In the global optimization step. **210**, a rough estimate of the net lengths are made (based on cell placement information), and this net length estimate serves as an input for performing structural improvements which can be made to the netlist. These structural improvements involve timing-driven re-mapping of the desired circuit and other optimizations which can change the logic structure of the circuit. Other optimizations include logic level reduction, logic duplication, and automatic test pattern generation (ATPG) based're-wiring.

In the global optimization step **210**, only a few of the cells are preferably re-assigned to other positions, and thus most of the cell assignments which have been performed during coarse placement are intact. After the global optimization step **210** is performed, the few re-assigned cells are placed in feasible buckets, while maintaining the sizes of the buckets relatively equal to each other. Preferably, according to the present invention, the global optimizations step **210** is performed before the global routing step **215**, since a rough interconnect model (resulting from the coarse placement step **205**) is typically sufficient when performing the global optimization step **210**. In addition, since the global routing step **215** has not yet been performed at this stage of the design process, there is no need to update any global routes after performing the global optimization step **210**.

At this stage of the design process, the following special nets may also be pre-routed, such as power and ground nets and clock nets. Commercially available routers typically have power net routing capabilities. The power net pre-routing is performed in full detail, and immediately written in the chip image database. The structure of the power nets may, for example, be a regular mesh of nets which are over-dimensioned to account for the fact that the actual power consumption will not be known until at a very late stage in the physical design process. Power estimation and simulation may be performed by an external tool such as PowerMill™ from the Epic Technology Group of Synopsys, Inc. The routing of clock nets is customer-specific. The clock nets are typically pre-routed in a fixed pattern, such as H-trees.

Global Routing

The global routing step **215** (FIG. **2**) is used to determine the net topologies (i.e., net lengths) and the interconnection layer assignments for the nets. The net topologies information provides additional net delay information for use in the optimization step **220** (FIG. **2**). Specifically, the net topologies information serves as an input in the delay analysis and repeater insertion steps, both of which may be performed in the electrical optimization step **220** (FIG. **2**). The net topologies information provide a fairly accurate model for estimating the delays in the desired circuit, since the capacitance of each net can be estimated based on net length. Gate size adjustments can be performed in optimization step **220** to meet timing constraints. Repeaters are inserted in long nets to optimize the circuit area and still permit the timing requirements to be met, as discussed in co-pending and commonly assigned U.S. Patent Application entitled "Timing Closure Methodology". As also discussed in further detail below, the net topologies information are further used to guide the detailed routing step **250** (FIG. **2**).

Additionally, the global routing step **215** includes the optimization goals of minimizing the total net length of the desired circuit and of spreading the net routing evenly across available routing areas on the chip so as to avoid congestion problems.

At this point in the design flow (global routing step **215**), the global router is preferably not "timing driven" in the sense that that timing violations in the desired circuit will be tolerated at this stage of the design process and subsequently compensated for in the optimization step **220**. In contrast, current methodologies perform timing-driven placement and timing-driven global routing in early stages of the design flow after the netlist input. By placing timing constraints on a significant amount (e.g., 30%) of global routes at this stage of the design process, less than optimum routing may result, since the imposed timing constraint may prohibit the routing of these nets as longer-length nets which would thus prevent routing congestion problems. Consequently, in conventional methodologies nets that have been determined and set cause over-concentration of cells in a particular area.

It is further noted that in the present invention, five (5) layers of interconnections (conducting medium) may be used. A minimum number of three (3) interconnection layers should at least be available, while the maximum number of interconnection layers may vary.

In the global routing step **215**, a conventional global router may be used. The router determines the net topologies based on the cell placement information obtained from coarse placement step **205**. Unlike conventional approaches, according to the present invention, the global router need not have the "exact" positions of cells to perform the global routing. Global routing is performed based on cell placement information obtained from the coarse placement step **205**.

FIG. **6A** is a partial top view of a first interconnection layer **450** to be formed in the semiconductor die, while FIG. **6B** is partial top view of a second interconnection layer **455** to be formed in the same die. The global routes formed during the global routing step **215** include a set of connected net segments, generally illustrated as **460**. According to one preferred embodiment of the present invention, each net segment **460** starts at a middle region of a bucket **310** and ends at the middle region of another bucket **310**. For example, assuming the interconnection layer **450** in FIG. **6A** is a metal-1 layer, the net segment **460***a* begins at the middle region of the bucket **310***a* and ends at the middle region of the bucket **310***b*. Assuming the interconnection layer **455** in FIG. **6B** is a metal-2 layer, a net segment **460***b* begins at the middle region of the bucket **310***c* and ends at the middle region of the bucket **310***d*. As further shown in FIGS. **6A** and

6B, each net segment **460** is straight and is further assigned to.a particular interconnection layer (e.g., interconnection layers **450** or **460**).

As discussed above, when the global routing step **215** is performed, the actual positions of the cells to be placed (and cell terminals) are not exactly known. Thus, each net segment that spans across bucket boundaries is approximated by being configured to start at a middle region of a bucket and end at a middle region of another bucket, for estimation purposes.

The global router can be programmed to read and take into account the pre-routed nets by decreasing the available routing resources along the paths of the pre-routed nets.

Electrical Circuit Optimizations

At this point in the design process, shown in FIG. **2** as step **220**, the approximate topologies of all nets that cross bucket boundaries are known based on the global routes. These net topologies provide a fairly accurate model for estimating the delays of these nets. The estimated delay of a given net is determined based on the net capacitive load approximations as determined by net length. These estimated. delays can then be compared with the timing requirements (which were determined prior to the coarse placement step **205**), so as to determine if the timing requirements will be met.

To meet the predetermined timing requirements (constraints), adjustment of the cell sizes and net wire widths can be made. Cell size adjustment is the process of selecting the appropriate drive strength for each cell such that the timing requirements are met. The drive strengths are selected directly from the library or by duplication of library cells. A wider range of cell sizes can also be created if the outputs of cells are tied together. For example, from a cell library with cells having the drive strengths "1", "2", and "4", a cell with a drive strength of "3" can be created by connecting in parallel a cell with drive strength 1 with a cell with drive strength 2.

Net wire width adjustments can also be used in conjunction with cell sizing to satisfy predetermined timing requirements for critical nets in the desired circuit wherein the critical nets are defined as nets close to their capacitance budgets. In other words, the timing parameter of a given global route may be set based on net wire width adjustments. As discussed below, the timing constraint for a given net is met if the net is within its capacitance budget. Net wire width can be adjusted to a discrete width having a value as shown by equation (1):

$$\text{Net Width}=(\text{minimum net width})\times N \quad (1)$$

wherein the minimum net width is typically 0.4 micro-meter and N=1.0, 1.5, 2.0, 2.5, etc. The range for the parameter N depends on the adjusting capability of the router.

Net wire width adjustments serve to assist in satisfying electro-migration constraints in design rules and timing constraints. An electro-migration constraint is the maximum current value which is permitted to flow through a wire of a certain width so that the wire does not result in an early burn out. The delay of a given net is represented by π=RC wherein R is the net resistance and C is the net capacitance. Thus, the delay of a given net can be adjusted by changing the net resistance R and/or net capacitance C values. Referring now to FIG. **7**, there is seen a net **470** having the length L and the width W. By increasing the net length L (and holding constant the net width W and net capacitance C), the resistance R of the net **470** increases. By increasing the net width W (and holding constant the net length L and net capacitance C), the resistance R of the net **470** decreases.

As an option in the electrical optimization step **220**, buffers can also be inserted at proper points in the topologies of long nets to save area. If a buffer is to be inserted, then the netlist and the global route will change. Co-pending and commonly-owned U.S. Patent Application entitled, "Timing Closure Methodology," describes a methodology for performing buffer insertions in order to save area.

At this stage of the design process, it is known whether the timing requirements can be met. If the timing requirements can be met, then the timing budget on each of the nets is fixed. The subsequent steps **225** to **254** are performed so that the timing requirements are satisfied. To insure satisfaction of the timing requirements, the circuit area may be enlarged.

Net Capacitance Budgeting

Net capacitance budgeting step **225** follows the optimization step **220**, and the timing requirements for the topologies of the nets related to global routes are fixed and will have to be met. As known to those skilled in the art, the delay provided by a given net is determined by the resistance of the net and the capacitance between the given net and a neighboring net. It is difficult to adjust the resistance of a given net, and as discussed above, net resistance adjustment involves adjustment of the net wire width. Thus, to adjust the delay provided by a given net so as to meet timing constraints, the capacitance between the given net and a neighboring net can be adjusted by controlling the distance between both nets. Therefore, the adjustments of capacitance between nets provide another method for setting the timing parameters of global routes.

In the capacitance budgeting step **225** (FIG. **2**), the maximum amount of capacitance is determined for each of the nets associated with global routes such that timing constraints for those nets are met. Referring now to FIG. **8A**, there is seen a net **500** comprising the segments **502**, **504**, and **506** which resulted from the global routing step **215**. A via **508** is disposed between the intersection of the segments **502** and **504**, while a via **510** is disposed between the intersection of the segments **504** and **506**. The Elmore delay model is preferably used to determine a maximum capacitance value for each net segment in the designed circuit, such that the designed circuit will likely meet the timing constraints. This maximum capacitance is also known as the "capacitance budget" which denotes the total capacitance (including net-to-ground capacitance and net-to-net capacitance).

Different variations based on the Elmore delay model may be used to determine the capacitance budget of a given net segment. For example and as best illustrated in FIG. **8B**, the segments **502**, **504**, and **506** can be modeled as discrete resistive and capacitive elements. The resistor R_{502} represent the resistance of the segment **502**, while the capacitor C_{502} represent the capacitance of the segment **502**. Similarly, the segment **504** is modeled into the resistor R_{504} and capacitor C_{504}, while the segment **506** is modeled into the resistor R_{506} and capacitor C_{506}. The values of the capacitors C_{502}, C_{504} and C_{506} can be set such that the delay constraint for the net **500** is met.

The Elmore delay based model of FIG. **8B** can be varied by taking into account the resistance provided by the vias in the net segment intersections. For example and as best illustrated in FIG. **8C**, the resistors R_{508} and R_{510} have been added to represent the resistance provided by the vias **508** and **510**, respectively. Other variations can be made on the Elmore delay based models shown in FIGS. **8A** and **8B**. Additionally, in measuring the actual delay provided by the resistance and capacitance of a given net or net segment, a conventional timing tool is used.

FIG. **8D** shows an example wherein two given adjacent net segments **506** and **520** are separated by the distance **515**.

The capacitance between the two given adjacent net segments **506** and **520** decreases as the distance **515** between the net segments increases. Conversely, the capacitance between the two given adjacent net segments **506** and **520** increases as the distance **515** between the net segments decreases. This distance **515** is controlled in the track routing step **230** (FIG. **2**), during which the positions of the global routes are fixed. Thus, the present invention permits capacitance control on a net-per-net basis.

In contrast, conventional methodologies typically attempt to decrease net delay by routing the nets based on a "worst case capacitance" analysis. This approach leads to larger die sizes than necessary and greater power consumption requirements.

Track Routing

In the track routing step **230** (FIG. **2**), which follows the capacitance budgeting step **225**, the routing order for the global routes are determined. The track routing step **230** further determines the actual positions in which the global routes will be fixed. The actual positions of the global routes are fixed such that the capacitance between nets are set at values which permit the timing requirements of the designed circuit to be met.

Net neighbor relations can be determined at this stage of the design process, while a fairly global view of the circuit is still possible. Net neighbor relations are important since they may also determine cross-talk sensitivity. At the simplest level, cross-talk constraints can be translated into net neighbor constraints.

Preferably, the track router operates substantially in the upper layers (which are the metal-3, metal-4 and above layers), thereby leaving the metal-1 and metal-2 layers for the detailed router to make local connections.

FIG. **9**A is a partial top view of an interconnection layer **550** in a semiconductor die, as shown during a horizontal track routing step, while FIG. **10**B is partial top view of the same interconnection layer **550** during a vertical track routing step. The track router fixes the final position of each net segment **460** (wherein a net segment **460** spans across at least a bucket boundary and may span several routing buckets **310**).

In contrast, conventional design approaches do not use a track routing step before performing the step of fixing the positions of each of the cells in the circuit. Conventionally, cell placement is performed, and global routing and detailed routing steps follow. In addition, in conventional approaches, a pin assignment step is typically required whereby internal pins are placed in the boundaries (interface) of local routing regions so that the router is able to perform local routing. In contrast, the present invention does not require the conventional pin assignment step since the track routing step **230** implicitly accomplishes pin assignments between buckets. When the track router places entire net (wire) segments across several buckets, pin positions are also fixed automatically on the bucket boundaries which are crossed by the net segments.

In accordance with the present invention, the track router operates based on defined channels (such as channel **560** in FIG. **9**A and channel **565** in FIG. **9**B) across the chip as the wire segments **560** are fixed. A channel is a defined region spanning the width of the integrated circuit design. A channel may span over as much as about one-hundred (100) buckets. The height of a channel is equal to about one (1) bucket.

During the horizontal track routing step (as best shown in FIG. **9**A), the track router fixes the position of each of the generally horizontal net segments **460**H along given horizontal channels (e.g., horizontal channel **560**). During the vertical track routing step (as best shown in FIG. **9**B), the track router fixes the position of each of the generally vertical net segments **460**V along given vertical channels (e.g., vertical channel **565**). The positions of the generally horizontal net segments **460**H and generally vertical net segments **460**V are set so that the capacitance budgets (as determined in step **225** in FIG. **2**) are met.

The track router works its way from one channel to another channel, and works on channels in all of the interconnection layers. The track router places net segments **460** based on the global router's determination of which of the net segments **460** should pass through a given channel, wherein the width and the maximum capacitance of each net segment **460** have also been determined previously.

The ordering and spacing of the net segments **460** will have the most significant influence on the capacitive load on the nets. The reason is that the capacitance between two nets (or net segments **460**) increases as the distance between the nets decreases. For example, FIG. **10** shows a channel **600** wherein the spacing between the net segments **460***c* and **460***d* have been set to meet the capacitance budgets.

The track router will preferably place the net segments **460** in a channel (e.g., channel **600**), while optimizing the ordering of the placing such that the capacitance budget of each net segment is met. To allow maximum freedom for the ordering of the placement of net segments **460**, vertical constraints should be eliminated, wherein elimination of vertical constraints means that a net ending in a bucket **310** occupies an entire row of the bucket **310**. As known to those skilled in the art, by eliminating vertical constraints the routing of a given channel will not depend on the routing which has been performed on an adjacent channel. The elimination of vertical constraints thus removes any requirement that the channels (e.g., channel **600**) be routed in a particular order, for each interconnection layer. The elimination of vertical constraints allows the designer more flexibility in dealing with cross-talk issues in the design.

FIG. **10** shows an example wherein vertical constraints have been eliminated. The net segment **460***e* will occupy its given row **605** in the buckets **310***e*, **310***f*, and **310***g*. The net segment **460***g* will occupy its given row **610** in the buckets **310***g* and **310***h*. By eliminating the vertical constraints, empty spaces will be present in some of the buckets **610**. For example, the net segment **460***e* does not completely extend across the bucket **310***g*. Since the vertical constraints are eliminated in the channel **600**, it is not permissible to move the net segment **460***f* into the row **605** (of buckets **310***g*, **310***h*, **310***i*).

The end position **615** of the net segment **460***e* is not exactly known at this stage of the design process, and therefore, as described previously, is approximated as ending in the middle of the bucket for the track routing step. The end position **615** of the net segment **460***e* may be finalized during the vertical track routing if the global route "changes directions". For example, in FIG. **9**B the end position of a horizontal net segment **460**' is determined when it changes directions upon formation of the vertical net segment **460**" during vertical track routing. The end position **615** of the net segment **460***e* may also be finalized by the detailed router during the detailed routing step **245** (FIG. **2**) if the net segment **460***e* is connected to a "cell terminal" in the bucket **310***g*.

FIG. **11**A is a partial top view of an interconnection layer **650** in a semiconductor die wherein the track router is unable to place all of the net segments **655** in a channel **660**. In this example, the net segment **665**A will not fit in the channel

660. Some net segments may not fit in their assigned channel if, for example, there are many unfriendly pairs of net segments associated to a channel, thereby forcing the net segments to be spaced at a greater distance. It is difficult to predict the effect that net "unfriendliness" will have on spacing between the net segments.

If certain net segments **655** (such as net segment **665A** in FIG. **11A**) do not fit, then all buckets **310***j* in the channel **660** are enlarged to create additional space to fit such net segments **665**. In FIG. liB, for example, the buckets **310***j* in the channel **660** have been extended in the direction of arrow **670** so that net **665A** will fit within the intended bucket. As discussed below, bucket enlargement is performed during the bucket equalization step **232** (FIG. **2**).

Bucket Equalization

Prior to the electrical optimization step **220** (FIG. **2**), cells are associated to the buckets **310** by the coarse placer wherein the sizes of all buckets **310** are equal. This is shown in FIG. **12A** wherein an interconnection layer region **700** includes equal sized buckets **310**. During the electrical optimization step **220** and/or track routing step **230** (FIG. **2**), the contents assigned to some buckets may change or the sizes and relative positions of the contents may change. Thus some of the bucket sizes may also change. As stated previously in step **220** (FIG. **2**), electrical optimization may lead to an increase in the sizes of some cells which are assigned to given buckets. As also stated above, the widths of and the spacing between net segments may change due to capacitance budgets or electrical constraints. As a result some buckets may need to be increased, and thus some buckets may become larger in size than other buckets, and this is shown as bucket equalization step **235** in FIG. **2**. Typically, the size of a given bucket which has increased in size will not exceed another bucket in size by more than about 5%. Ideally, however, all buckets should essentially have the same size, since the total layout size of the integrated circuit will be determined by the largest bucket in each of the rows (or in each of the columns).

FIG. **12B** shows a bucket array wherein some of the bucket sizes are changed, resulting in a larger circuit. If the bucket **310***k* is the largest bucket in the row **710**, it will determine the size (width) of the row **710**. If the bucket **310***l* is the largest bucket in the column **720**, it will determine the size (width) of the column **720**. Thus, the largest bucket in a given row (or given column) will have an impact on the size of the chip.

The largest bucket (or buckets) in a given row (or given column) may include a "critical area", which is shown in FIG. **12C** as area **735**, which critical area is the difference in area between that bucket and the next largest bucket in a row or column direction. The bucket equalization step **235** minimizes the critical areas in the bucket array so as to optimize the size of the chip. The result of the bucket equalization step **235** is shown in FIG. **12C**. The "overflow cells" are moved (assigned) next to neighboring cells so as to reduce the size of the largest bucket in a given row **710** (or given column **720**), thereby minimizing the critical area.

Preferably, only "non-critical" cells are moved to a given neighboring bucket, such that the electrical characteristics of the nets will not change or will change as little as possible. Non-critical cells are cells which can be moved without increasing the associated net lengths in a manner that substantially adversely affects timing constraints. A non-critical cell may also drive a given net which is within the net capacitance budget. The non-critical cell can be moved, since the resulting increase in net length still falls within the net capacitance budget. If the electrical characteristics of the altered nets are kept within the capacitance timing budgets, it will be unnecessary to again perform the electrical optimization step **220** (FIG. **2**). Re-performing the electrical optimization step **220** would lead to an undesirable loop, since this would change the track routing criteria performed in the track routing step **230**.

Additionally, a greedy algorithm may be used when performing the bucket equalization step **235**.

Re-Perform the Steps of Global Routing, Net Capacitance Budgeting, and Track Routing

The bucket equalization step **235** (FIG. **2**) will change the position of cells, thereby changing the global routing and detailed routing data. As a result, the global routing and track routing steps will have to be re-performed. The global routing step is again performed (step **240** in FIG. **2**) to incrementally change the circuit and is subject to the following constraints. First, only the topologies of the affected "non-critical" nets are changed, wherein a non-critical net is one which falls within its capacitance budget. Second, all nets remain within their strict capacity criteria. These capacities are the net capacitive load requirements so that the total net topology is likely to meet the timing requirements of the circuit (wherein timing requirements have been fixed in step **220** of FIG. **2**). The above constraints minimize the extent of the changes when global routing is again performed. Thus, the topology of most nets in the circuit will remain unchanged.

Additionally, the global routing step **240** (as well as the track routing step **230**) are driven by the timing constraints which have been established in the optimization step **220**. Thus, global routing is performed so that the net capacitive load requirements are met, thereby permitting circuit timing requirements to be met.

On the other hand, after the net capacitance budgeting step **225** is re-performed, the track routing step **230** is completely re-performed, as it is not possible to incrementally change the track routing of the circuit.

Detailed Placement

Unlike conventional design processes, in the present invention the detailed placement step **245** (FIG. **2**) is performed after the track routing step **230**. Thus, the invention permits the option of adjusting the placement of cells based on where the global routes are placed. In contrast, under conventional approaches, the net routing typically needs adjustments based on cell placement results.

Under the detailed placement step **245** (FIG. **2**), standard cells are assigned final positions. FIG. **13** shows an example of a placement **800** in a portion of a bucket **310***m* wherein the cells, generally shown as **805**, are placed close to the net segments **810** and **815** and are oriented appropriately in the bucket portion. General placement techniques known those skilled in the art are used. Such general placement techniques for placing cells in the bucket include min-cut, simulated annealing, and quadratic placement.

The detailed placement step **245** can be performed by a detailed placer based on a conventional algorithmic engine. The detailed placer places cells **805** in one bucket at a time, and may work on two adjacent bucket portions so that proper connections (between cells **805** in adjacent bucket regions) will result from the design process. One objective of the detail placer is to minimize the total net length in the circuit and to possibly minimize the length of a limited set of "critical nets" wherein the critical nets are defined as nets close to their capacitance budgets. Preferably, the detailed placer is also subject to a "net alignment constraint" whereby the detailed placement is adapted based on the track routing step **230** (FIG. **2**) output. In other words, the

detailed placement of cells is guided by the positions of the global routes as fixed in the. track routing step **230**. Net alignment constraints can be taken into account by use of the simulated annealing method. In contrast, conventional approaches attempt to complete the global routes based on the results of the detailed placement.

Pre-placed cells can be taken into account during detailed placement by de-activating certain resources available to the detailed placer. Simulated annealing can take into account the pre-placed cells.

The detailed placement step **245** is performed after the track routing step **230** for the following reasons. First, it is more important to control the net capacitance than to minimize the area of the integrated circuit. Performing the placement step **245** after the track routing step **230** will lessen the difficulty in controlling the net capacitance (since final placement does not yet exist as track routing is performed). Second, as discussed above, the track routing step **230** may unpredictably stretch a row (or columns) of buckets, thereby changing the bucket shapes and sizes (see, e.g., FIG. **11**A–**11**B). In contrast, the detailed placer has very predictable limits and will make changes which will not require the re-performing of global routing (as in step **240**). Third, iteration will be much faster in the bucket equalization loop (steps **232**, **235**, **240**, **225** and **230**) if the final placement does not yet exist in the circuit. A faster bucket equalization loop lead to a faster design process. Fourth, the detailed placement step **245** can be performed in a very small "window" which is smaller than the track routing step **230** "window" wherein a detailed placement window is a single bucket. This has an advantage of enabling a faster design process.

Reference is now made to FIG. **14** wherein a particular bucket **310** has standard cells which are generally shown as **850**. The bucket **310***n* has clearly defined boundaries on the bottom side **855** and top side **860**. The bottom side **855** and the top side **860** coincides with the horizontal power lines of the circuit. Since some standard cells **850** have variable widths, some cells may overlap the left side **865** and/or right side **870**. In the example of FIG. **14**, the cells **850***a* and **850***b* unavoidably overlap the right side **870** and into the neighboring bucket **310***p*. This overlapping of cells is not a problem if there is never a stretching of the vertical channels on the left or right sides **865** and **870**. Pushing a global pin (cell terminal) over the boundary could invalidate the track and global routing and should therefore be preferably avoided.

Detailed Routing

In step **250** (FIG. **2**), detailed routing is performed by a detailed router which has a primary objective of achieving routing completion of the circuit and a secondary objective of minimizing any additional load on the critical nets (i.e., nets close to their capacitance budgets). The detailed router generates the local wiring in a given bucket (or a small collection of given buckets). Since the cell sizes and positions have been fixed during the detailed placement step **245** (FIG. **2**), the integrated circuit's performance will depend on the detailed routing of step **250**.

The detailed router generates the local wiring in each bucket (or in a small collection of buckets) based on its reading of existing routes (e.g., pre-routes and net segments formed during the track routing **230**) and of the net list. The output of a detailed router is a mask pattern suitable for a specific VLSI fabrication technology. The detailed routing step **250** will preferably use a grid-based approach. Grid-based routers are simple, sturdy, tested and can enforce a desirable vertical alignment between the interconnection layers. An example of a grid-based router is the conventional Lee-type maze router. The Lee-type maze router is an "over-the-cell" router which uses all available routing resources. The router must be adapted to handle the three-dimensional fine grid. This is accomplished by adapting the wave-front expansion in the Lee router engine.

Although the minimum spacing and width requirements for nets might not be uniform over the different interconnection layers, it can be assumed that a fine grid still allows efficient routing in all of the interconnection layers, since the process design rules are generally targeted towards a grid. In other words, existing design rules are adapted so that a single fine grid can be applied to all of the interconnection layers, thereby permitting the vertical alignment of grids for each layer.

Preferably, the router is adapted to handle a three-dimensional fine grid (e.g., ½ of the normal track pitch (wherein the track pitch is the distance between adjacent tracks **915**, measured in a direction perpendicular to the tracks **915**). By doubling (or increasing) the resolution of the grid, the relevant deep sub-micron requirements of the integrated circuit can be incorporated. These deep sub-micron requirements relate to, for example, minimizing cross-talk between nets. As stated previously, cross-talk is caused by the capacitive coupling between neighboring nets.

In FIG. **15**, the grid is represented by the lines **900** and the nets are generally shown as **902**. In a 0.25 micron CMOS process, the minimum routing pitch (grid spacing) is about 1.0 micron. The use a fine grid allows the detailed router to vary the net spacing (represented by arrows **905**) and the net widths (represented by the arrows **910**). The net spacing and width can be varied without use of a whole track which generally shown as **915**. This approach does not require extra memory, as the encoding per grid point can be reduced.

The net resistance can be varied by adjusting the net width **910**. The net resistance decreases linearly as the width **902** increases. The net capacitance can be varied by adjusting the distance **905** between the nets **902**. The capacitance of a net decreases linearly (as an approximation) as the distance **905** increases. To fine tune the resistance and capacitance of a net, it is necessary to have a fast and reasonably accurate net extraction capability. The net width and capacitance can be passed as constraints to the detailed router.

It is an option that the detailed router be "capacitance driven" wherein the capacitance-driven router takes into account the maximum capacitive load of a net before generation of the net. This option may not be required since the integrated circuit is already configured to meet the capacitance load requirements in the track routing step **230** (FIG. **2**).

The routing area in a bucket is small, for example, 10×10 cells, which is about 200×200×5=200,000 fine grid points.

Routing Completion

If routing completion is achieved, then the layout is generated as an output in step **255** (FIG. **2**). However, if the detailed router is unable to achieve routing completion, the following options are available for the circuit designer for routing completion (step **254** in FIG. **2**). First, routing completion can be attempted by the router by use of "rip-up-and-reroute strategies". Second, existing nets can be "pushed aside" to correct for wrong decisions made earlier in the design process which led to the incomplete routing.

As a third option for achieving routing completion, the detailed placement step **245** can be re-performed such that the new placement will enable routing completion. According to the present invention, this option can be performed quickly and efficiently, since the placement of cells occurred

after the global routing step (**215** or **240**) and after the track routing step **230**. In contrast, in conventional approaches, the global routing step occurs after the cells are placed (see, e.g., U.S. Pat. No. 5,483,461, issued to Lee et al. on Jan. 9, 1996). Thus, in conventional approaches, by re-performing the cell placement step, the global routing step will have to be performed again.

With reference to FIG. **16A**, there is seen another method for achieving routing completion and for meeting capacitance constraints by use of tear lines (generally illustrated as horizontal tear lines **950***h* and vertical tear lines **950***v*). The tear lines are torn in a chip layout **955** by a conventional chip tear line tool such that additional routing spaces are created in the chip. To assist in describing this feature of the invention, only one horizontal tear line **950***h* is shown to span each bucket **310**. As best illustrated in FIG. **17**, however, a plurality of tear lines **950***h* may span a given bucket, since a chip layout may contain natural tear lines coinciding with the horizontal power nets.

Referring back to FIGS. **16A–16C**, assume that the buckets **310***q* and **310***r* initially have incomplete routing. In FIG. **16B**, a chip layout **955** is torn open along the horizontal tear line **950***h* which is across the bucket **310***q*. This creates the additional space **960** which will enlarge all buckets **310** in the row **965**, including the bucket **310***q*. Nets which will complete the routing in the bucket **310***q* are placed in the additional space **960**. The routing and layout of the buckets, which are not in the row **965**, are unaffected by the creation of the additional space **960**. Similarly, in FIG. **16C** the chip layout **955** is torn open along the vertical tear line **950***v* which is across the bucket **310***r*. This creates the additional space **970** which will enlarge all buckets **310** in the column **975**, including the bucket **310***r*, and nets for achieving routing completion can be placed in the additional space **970**

As shown in FIGS. **16B** and **16C**, tearing open the chip layout along a tear line will enlarge all buckets in a row (or column). Although tearing open the chip layout will increase the area of the integrated circuit, the created additional space (e.g., space **960**) is utilized for routing completion. In contrast, the conventional methods of achieving routing completion not only enlarges the area of the circuit, but also the additional area added are in portions which are not required in the circuit.

Reference is now made to FIG. **17** which shows a bucket **310***s* for the purpose of illustrating the method for achieving routing completion by use of tear lines. According to the present invention, pins (cell terminals), generally illustrated as **1000**, are placed adjacent to a given tear line **950***h*. All pins **1000** are connected to the nearest tear line **950***h* by using an "escape path" (generally illustrated as **1005**) wherein an escape path is a temporary wire or conductor which connects a given cell terminal **1000** to two horizontal tear lines **950***h* which are adjacent the given cell terminal **1000**. The escape paths **1005** are removed as soon as a net is routed over the cell terminal, thereby achieving routing completion.

Referring now to FIG. **18**, assume that a given cell **1010** has a terminal **1000** which can be coupled to a given tear line **950***h* via escape path **1005** and that routing completion has not been achieved for the terminal **1000**. Assuming further that the chip layout is torn open at the tear line **950***h* to create additional space **1020**. At least one net **1025** may be placed in the additional space **1020**, and the terminal **1000** may be coupled to the net **1025** to achieve routing completion.

FIG. **19** is a snapshot of a routing grid **1050** at the intersection of two tear lines (horizontal tear line **1055** and vertical tear line **1060**). The routing grid **1050** is the "in-core" representation of the routing environment in a chip layout, and is a three-dimensional near-regular mesh of nodes (generally represented by the dots **1065**). The horizontal edges **1070** and the vertical edges **1075** represent segments of nets. The edges **1080** (along the z-direction) represent vias for connecting a net in one interconnection layer to another net in structure can be hidden from the detailed router by embedding the tear line structure in the routing grid as best shown in FIG. **19**. The horizontal tear lines are crossed by vertical layers and vertical tear lines are crossed by horizontal layers.

The three-dimensional fine grid **1050** in FIG. **19** is configured so as to minimize the routing of the detailed routes along the tear lines of the chip layout. The cost factors are tuned such that routing along the tear lines **1055** and **1060** will be extremely expensive. If the cost of an edge (e.g., tear line) is high, then the detailed router will unlikely use the edge. If routing completion is not achieved, then the detailed router will consider the use of the tear line so that the chip layout is torn open to provide additional spaces for routing. Thus most nets will be routed over the cells. If a given connection cannot be made, then the detailed router will consider using the tear line for routing completion.

While it is preferred to stay within the timing constraints during steps **225–250**, it is within the scope of the present invention to go beyond the timing constraints and to subsequently determine whether to correct the circuit later in the design process to meet the timing constraints (as fixed in step **220**).

The integrated circuit chip resulting from the method of the present invention may have, for example, 1,000,000 cells and 1,000,000 nets. Each cell measures 10x10 routing tracks and has three inputs and one output. In one preferred embodiment of the invention, the chip is configured in an array of 1,000x1,000 cells which, in turn, translates to 10,000x10,000 tracks (or 1x1 cm). In addition, a row-based standard cell style layout is preferably implemented.

This specification shows the components (e.g., buffers, cells, nets) which exist when the integrated circuit layout is produced. However, it is realized by those skilled in the art that when performing the steps in accordance with the present invention prior to producing the layout, the shown components are data representation stored in the computer and not the actual devices.

Thus, while the present invention has been described herein with reference to particular embodiments thereof, a latitude of modification, various changes and substitution are intended in the foregoing disclosure, and it will be appreciated that in some instances some features of the invention will be employed without a corresponding use of other features without departing from the scope of the invention as set forth.

What is claimed is:

1. An automated method for designing an integrated circuit layout with a computer, based upon an electronic circuit description and upon a selected plurality of cells from a cell library, comprising the steps of:

(a) assigning each of the cells to one of a plurality of buckets designated on the integrated circuit layout, each of the cells being connected to one of the other cells;

(b) performing global routing to connect at least some of the selected cells of step (a) together such that global routes are formed to provide net topology information;

(c) performing track routing which sets the position of each of the global routes;

(d) performing detailed placement.such that the positions of all selected cells are fixed within each of the buckets designated on the integrated circuit layout; and

(e) performing detailed routing such that detailed routes are formed to complete the integrated circuit layout.

2. The automated method of claim 1 further comprising:

prior to the step (b) of performing global routing, performing global optimization of cell sizes based upon the assignment of each of the cells of step (a).

3. The automated method of claim 1 wherein the step of performing global optimization includes reducing a logic structure formed by at least some the cells of step (a).

4. The automated method of claim 1 further comprising:

performing electrical optimization based upon the net topology information of step (b) to satisfy predetermined timing constraints for at least some of the global routes of step (b).

5. The automated method of claim 4 wherein the step of performing electrical optimization comprises:

adjusting the width of at least some of the global routes of step (b).

6. The automated method of claim 4 wherein the step of adjusting the width of at least some of the global route satisfies electromagnetic constraints of said some of the global routes.

7. The automated method of claim 4 further comprising:

after the step of performing electrical optimization, determining a capacitance load for each of the global routes of step (b) so that the capacitance load of each of the global routes satisfies predetermined timing constraints associated with each of the global routes.

8. The automated method of claim 7 wherein the step (c) of performing track routing sets the capacitance load of each of the global routes of step (b).

9. The automated method of claim 8 wherein the capacitive load for a certain one of the global routes is used to determine the location of another global route.

10. The automated method of claim 7 wherein the step (c) of performing track routing sets the position of each of the global routes to meet predetermined timing constraints associated with each of the global routes.

11. The automated method of claim 1 further comprising:

prior to the step (d) of performing detailed placement, performing bucket equalization to minimize an area of the integrated circuit layout.

12. The automated method of claim 11 wherein the step of performing bucket equalization comprises:

moving one of the cells of step (a) to further reduce the area of the integrated circuit layout.

13. The automated method of claim 12 where the step of moving one of the cells assists in satisfying predetermined timing constraints associated with at least some of the global routes of step (b).

14. The automated method of claim 12 further comprising:

after the step of performing bucket equalization, performing a timing-driven global routing to adjust at least some of the global routes which are connected to the moved cell.

15. The automated method of claim 14 further comprising:

after the step of performing the timing-driven global routing, performing track routing to determine the positions of the global routes of step (b).

16. The automated method of claim 1 further comprising:

achieving routing completion to connect one of the cells in one of the buckets to another one of the cells in the same bucket.

17. The automated method of claim 16 wherein the step of achieving routing completion comprises:

repeating the step (d) of performing detailed placement such that at least some of the cells permit routing completion.

18. The automated method of claim 16 wherein the step of achieving routing completion comprises:

forming a tear line in the chip layout; and

disposing a net in the tear line; and

forming a conductor for coupling one of the cells in one of the buckets to the net disposed in the tear line.

19. The automated method of claim 1 further comprising:

associating a hierarchical module in one of the buckets of step (a).

20. The automated method of claim 1 wherein the number of cells associated within one of the buckets of step (a) ranges from 20 to 200.

21. The automated method of claim 1 further comprising:

providing a hierarchical netlist;

tagging a selected intermediate hierarchical module in the hierarchical netlist;

flattening the hierarchical netlist while preserving the selected intermediate hierarchical module; and

generating a netlist for the step (a) of performing coarse placement if the selected intermediate hierarchical module fits within one of the buckets of step (a).

22. The automated method of claim 21 further comprising:

if the selected intermediate hierarchical module is larger than one of the buckets of step (a), partitioning the selected intermediate hierarchical module into smaller parts; and

selecting wires for coupling the smaller parts together so as to place each of the smaller parts adjacent each other.

23. The automated method of claim 1 wherein the step (a) of performing coarse placement permits violations of predetermined timing constraints associated with the global routes.

24. The automated method of claim 1 wherein the step (d) of performing detailed placement is performed after the step (c) of performing track routing.

25. The automated method of claim 1 wherein the step (b) of performing global routing placement permits violations of predetermined timing constraints associated with the global routes.

26. The automated method of claim 1 wherein the step (c) of track routing includes:

fixing the positions of a first selected plurality of global routes in a generally horizontal direction.

27. The automated method of claim 1 wherein the step (c) of track routing further includes:

fixing the positions of a second selected plurality of global routes in a generally vertical direction.

28. The automated method of claim 1 wherein the final positions for all cells of step (d) is based upon the position of each of the global routes as set in the step (c) of performing track routing.

29. The automated method of claim 1 wherein the step (e) of performing detailed routing generates detailed routes for coupling the cells to the global routes.

30. The automated method of claim 1 wherein the step (e) of performing detailed routing is performed by a maze router adapted for operation on a three-dimensional fine grid.

31. The automated method of claim 30 wherein the resolution of the three-dimensional fine grid can be increased to satisfy deep submicron requirements.

32. The automated method of claim 31 wherein the resolution of the three-dimensional fine grid is adjusted to control the capacitance between the detailed routes.

33. The automated method of claim 32 wherein the three-dimensional fine grid is configured to minimize the routing of the detailed routes along a tear line of the integrated circuit layout.

34. The automated method of claim 1 wherein the step (d) of performing detailed routing generates local routing for each bucket of step (a).

35. The automated method of claim 1 wherein the position of at least some of the global routes are set without vertical constraints during the track routing step (c).

36. An integrated circuit layout produced in accordance with the automated method of claim 1.

37. A method for designing an integrated circuit layout by using a computer, based upon the selection of cells from a cell library, comprising the steps of:

(a) designating a plurality of buckets in the integrated circuit layout so that at least some of the cells are assigned in initial placements in the buckets;

(b) connecting a plurality of global routes having net lengths based upon the initial placements of the cells of step (a);

(c) performing track routing to fix the position of each of the global routes of step (b);

(d) performing detailed placement by placing additional cells in the buckets of step (a) after the fixing step (c) so that positions of selected cells are fixed within each of the buckets on the integrated circuit layout; and

(e) placing detailed routes to connect at least some of the additional cells together in one of the buckets.

38. The method of claim 37 further comprising:

performing optimization based upon the initial placements of step (a).

39. The method of claim 38 wherein the step of performing optimization comprises:

partially re-mapping at least some of the cells of step (a) to reduce a logic structure formed by cells of step (a).

40. The method of claim 39 further comprises:

after the step of performing electrical optimization, determining the capacitance budget for each of the global routes of step (b) so that each of the global routes satisfies predetermined timing constraints associated with each of the global routes.

41. The method of claim 37 further comprises:

after performing the connecting step (b), performing electrical optimization to satisfy predetermined timing constraints for at least some of the global routes of step (b).

42. The method of claim 41 wherein the step of performing electrical optimization comprises:

adjusting the widths of at least some of the global routes of step (b).

43. The method of claim 42 further comprising:

based on the step of moving the selected cell, performing global routing to incrementally adjust at least some of the global routes connected to the moved cell.

44. The method of claim 37 further comprising:

after performing the fixing step (c), moving a selected one of the cells in an incremental distance to minimize an area of the integrated circuit layout.

45. The method of claim 37 wherein the step (e) of placing detailed routes satisfies the predetermined timing constraints associated with each of the global routes of step (c).

46. The method of claim 37 wherein the detailed routes of step (e) are placed by a grid-based router.

47. The method of claim 46 wherein the grid-based router is adjusted to meet deep submicron requirements.

48. The method of claim 37 wherein the detailed routes of step (e) are placed to assist in satisfying capacitance load requirements associated for each of the detailed routes.

49. The method of claim 37 further comprising:

after the placing step (e), performing routing completion so as to complete the integrated circuit layout.

50. The method of claim 49 wherein the step of performing routing completion comprises:

re-performing the placing step (d) to generate another placement for the additional cells of step (d) to enable routing completion of the integrated circuit layout.

51. The method of claim 49 wherein the step of performing routing completion comprises:

forming a space in the integrated circuit layout for receiving a wire which enable routing completion.

52. The method of claim 51 wherein the additional space is a tear line in the integrated circuit layout.

53. The method of claim 51 further comprising:

forming a conductor for coupling a cell in one of the buckets of step (d) to the wire in the space formed in the chip layout.

54. The method of claim 37 wherein the initial placement of cells in step (a) permits timing violations in the global routes.

55. The method of claim 37 wherein the connecting step (b) permits timing violations in the global routes.

56. An integrated circuit layout produced in accordance with the method of claim 37.

57. A method for designing an integrated circuit layout, comprising the steps of:

(a) placing a first plurality of cells in initial positions designated by buckets;

(b) forming a plurality of global routes based upon the designated initial positions of the first plurality of cells of step (a);

(c) performing track routing to fix the positions of the global routes of step (b);

(d) performing detailed placement by placing a second plurality of cells based upon the positions of the global routes of step (c) such that the positions of all selected cells are fixed within the buckets designated on the integrated circuit layout; and

(e) placing a plurality of detailed routes to complete the integrated circuit layout.

58. The method of claim 57 further comprising:

performing optimization based upon the designated initial positions of the first plurality of cells of step (a).

59. The method of claim 57 further comprising:

after the fixing step (c), performing electrical optimization to satisfy the predetermined timing constraints of at least some of the global routes.

60. The method of claim 57 further comprising:

setting the capacitance of each of the global routes of step (b) prior to the fixing step (c).

61. The method of claim 57 further comprising:

after the fixing step (c), moving at least one of the first plurality of cells at an incremental distance to minimize an area of the integrated circuit layout.

62. The method of claim 61 further comprising:

revising a selected number of global routes coupled to a moved cell after performing the step of moving at least one of the cells.

63. The method of claim 62 wherein the revising step satisfies the predetermined timing constraints associated with at least some of the global routes.

64. The automated method of claim 57 wherein the placing step (e) is based upon a three-dimensional grid for guiding the detailed routes.

65. The automated method of claim 64 wherein the three-dimensional grid can incorporate deep submicron requirements of the integrated circuit layout.

66. The automated method of claim 57 wherein the fixing step (c) is driven by predetermined timing constraints associated with at least some of the global routes.

67. The automated method of claim 57 wherein the placing step (a) permits violations of predetermined timing constraints associated with each of the global routes.

68. The automated method of claim 57 wherein the forming step (b) permits violations of predetermined timing constraints associated with each of the global routes.

69. The automated method of claim 57 further comprising:

performing routing completion to form the integrated circuit layout.

70. The automated method of claim 69 wherein the step of performing routing completion comprises:

re-performing the placing step (d) so that routing completion is achieved after placement of the second plurality of cell.

71. The automated method of claim 69 wherein the step of performing routing completion comprises:

forming a space in the integrated circuit layout;

receiving a wires in the space; and coupling one of the second plurality of cell to the wire formed in the space.

72. The automated method of claim 71 wherein the space is formed by a tear line in the integrated circuit layout.

73. An integrated circuit layout produced in accordance with the automated method of claim 57.

74. An integrated circuit layout formed on a chip layout comprising:

a plurality of selected cells;

global routes for coupling the selected cells together, the global routes having been set in position by a track routing process;

detailed routes for coupling the global routes to the selected cells; and

a routing space formed in the chip layout for permitting routing completion of the integrated circuit layout;

wherein the cells are fixed within buckets on the integrated circuit layout by a detailed placement process.

75. The integrated circuit layout of claim 74 wherein the selected cells are sized so as to permit timing constraints of the integrated circuit layout to be met.

76. The integrated circuit layout of claim 74 wherein the global routes are routed so as to satisfy the timing parameters of the global routes.

77. The integrated circuit chip of claim 74 wherein the detailed routes are routed so as to satisfy the timing parameters of the global routes.

78. A timing-based integrated circuit layout formed on a chip layout comprising:

a plurality of cells;

timing-based global routes for coupling at least some of the cells together, the timing-based global routes being set in position by track routing so as to satisfy timing requirements of the integrated circuit layout; and

detailed routes for coupling the timing-based global routes to at least some of the cells, the at least some of the cells having been fixed in position within buckets designated on the integrated circuit layout by a detailed placement process.

79. An automated method for satisfying the timing requirements of an integrated circuit layout on a chip layout, comprising the steps of:

(a) placing a first selected plurality of cells in approximate positions in the chip layout;

(b) selecting global routes for coupling the first selected plurality of cells together;

(c) setting the positions of the global routes of step (b) using a track routing process such that a timing parameter of each of the global routes satisfies the timing requirements of the integrated circuit layout; and

(d) placing a second selected plurality of cells based upon the positions of the global routes of step (c) using a detailed placement process such that the positions of selected cells are fixed within each of buckets designated on the integrated circuit layout.

80. An integrated circuit layout produced in accordance with the automated method of claim 79.

81. An automated method for achieving routing completion in an integrated circuit layout, comprising the steps of:

(a) providing a first plurality of cells in approximate positions so that the cells are used to implement the integrated circuit layout;

(b) placing a plurality of global routes for coupling the first plurality of cells;

(c) fixing the positions of the global routes using a track routing process;

(d) placing a second plurality of cells based upon the positions of the global routes of step (c);

(e) performing detailed routing for coupling the second plurality of cells together and to the global routes; and

(f) performing routing completion so as to form the integrated circuit layout such that positions of all selected cells are fixed within buckets on the integrated circuit layout.

82. The automated method of claim 81 wherein the performing step (e) includes:

forming a tear in a chip layout; and

receiving a wire in the tear wherein the wire enables routing completion of the integrated circuit layout.

83. The automated method of claim 82 wherein the tear is formed by a chip tearing device.

84. An integrated circuit layout produced in accordance with the automated method of claim 81.

85. An automated method for designing an integrated circuit layout with a computer, based upon an electronic circuit description and by using a cell library containing a selected plurality of cells, comprising the steps of:

(a) based upon a portion of a computer program that contains a sequence of instructions, assigning each of the cells to one of a plurality of buckets designated on the integrated circuit layout, each of the cells being connected to one of the other cells based upon the electronic circuit description input to the computer;

(b) performing global routing to connect at least some of the selected cells of step (a) together such that global routes are formed to provide net topology information;

(c) performing track routing which sets the position of each of the global routes;

(d) performing detailed placement such that the positions of all selected cells are fixed within each of the buckets designated on the integrated circuit layout; and

(e) performing detailed routing such that detailed routes are formed to complete the integrated circuit layout.

86. A automated method for designing an integrated circuit layout by using a computer and based upon an electronic circuit description, comprising the steps of:

(a) using a portion of a computer program that contains a sequence of instructions, placing a first plurality of cells in designated initial positions;

(b) forming a plurality of global routes based upon the designated initial positions of the first plurality of cells of step (a);

(c) performing track routing to fix the positions of the global routes of step (b);

(d) performing detailed placement on a second plurality of cells based upon the positions of the global routes of step (c) such that positions of all selected cells are fixed in each of buckets designated on the integrated circuit layout; and

(e) placing a plurality of detailed routes to complete the integrated circuit layout.

* * * * *